OIPE
MAY 0 4 2000
PATENT & TRADEMARK OFFICE
JC9

169.1628

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	
PAUL RAYMOND HIGGINBOTTOM, ET, AL.	Examiner: Not Yet Assigned
Application No.: 09/523,595	Group Art Unit: Not Yet Assigned
Filed: March 10, 2000	
For: ENCODING METHOD AND APPARATUS	Date: May 3, 2000

Assistant Commissioner for Patents
Washington, D.C. 20231

CLAIM TO PRIORITY

Sir:

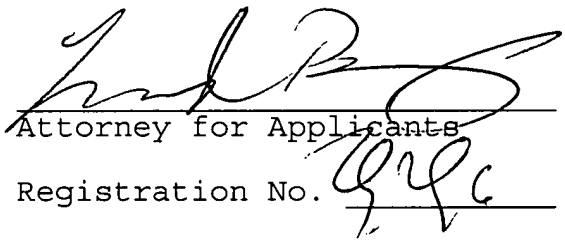
Applicants hereby claim priority under the International Convention and all rights to which they are entitled under 35 U.S.C. § 119 based upon the following Australian Provisional Application No. PP9186, filed on March 12, 1999.

A certified copy of the priority document is enclosed.

This Page Blank (uspto)

Applicants' undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All correspondence should continue to be directed to our address given below.

Respectfully submitted,



Attorney for Applicants

Registration No. 29,256

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

/pap

NY_MAIN 79885 v 1

This Page Blank (uspto)

169.1628

09/523595

Patent Office
Canberra

I, LEANNE MYNOTT, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PP 9186 for a patent by CANON KABUSHIKI KAISHA filed on 12 March 1999.

CERTIFIED COPY OF PRIORITY DOCUMENT

WITNESS my hand this
Fifteenth day of March 2000

LEANNE MYNOTT
TEAM LEADER EXAMINATION
SUPPORT AND SALES

This Page Blank (uspto)

S & F Ref: 447178

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Encoding Method and Appartus

Name and Address
of Applicant: Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, JAPAN
Name of Inventors: Eric Majani, James Philip Andrew and Paul Raymond Higginbottom.

This invention is best described in the following statement:

Encoding Method and Apparatus

Field of the Invention

The present invention relates to the field of digital image compression and in particular, to method and apparatus for performing a wavelet decomposition of an image.

Background to the Invention

The field of digital data compression and in particular digital image compression has attracted a great interest for some time. Most recently, compression schemes based on a Discrete Wavelet Transform (DWT) have become increasingly popular because the DWT offers a non-redundant heirachical decomposition of an image and resultant compression of the image provides favourable rate -distortion statistics.

Typically, the discrete wavelet transform (DWT) of an image is performed using a series of one-dimensional DWT's. A one-dimensional DWT of a signal (ie an image row) is performed by lowpass and highpass filtering the signal, and decimating each filtered signal by 2. Decimation by 2 means that only every second sample of the filtering processes is calculated and retained. When performing a convolution (filtering) the filter is moved along by two samples at a time, instead of the usual one sample. In this way for a signal of N samples there are N DWT samples: $N/2$ lowpass samples and $N/2$ highpass samples.

This is typically performed by buffering an entire image in Computer Memory such as Random Access Memory (RAM) and performing the DWT on the buffered image and then encoding the transformed image. Unfortunately, this approach requires a large amount of memory, particularly for image of 2000 pixels x 2000 pixels or more, and considerable memory bandwidth requirement if sufficient processing speed is to be attained.

Summary of the Invention

According to a first aspect of the invention there is provided a method of encoding an digital image comprising a plurality of pixels, said image being able to be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specified block size in number of coefficients, in a first and second dimension, the method comprising the steps of:

a) dividing the image into a plurality of tiles, each tile having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

b) selecting a current tile;

c) decomposing said current tile using the DWT to at least one level of decomposition to form a plurality of subbands including a LL, LH, HL and HH subband;

d) accumulating coefficients in each subband of the LH, HL and HH subbands to form blocks of said specified size and encoding each said block to a bit stream;

e) accumulating LL subband coefficients and repeating steps b) to e) until a predetermined number of coefficients of the LL subband have been accumulated;

f) assigning as a current tile said predetermined number of accumulated LL subband coefficients;

g) repeating steps c) to g) until the predetermined level of decomposition is reached; and

h) encoding the LL subband into the bit stream.

According to a second aspect of the invention there is provided a method of encoding an digital image comprising a plurality of pixels, said image being able to be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specifying a block size in number of coefficients, in a first and second dimension, the method comprising the steps of:

a) dividing the image into a plurality of tiles, each tile having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

b) selecting a tile of the image as a current tile;

c) decomposing said current tile using the DWT filter to a provide a plurality of

coefficients in a LL, LH, HL and HH subbands;

d) encoding coefficients of the LH, HL, HH subband of a current level into a bitstream;

e) determining if said current level is the predetermined level:

ea) encoding the coefficients of the LL subband into the bitstream, if the current level is the predetermined level, and repeating steps b) to e); and

eb) storing coefficients not previously encoded of the LL subband if the current level is not the predetermined level;

f) determine if the number of said stored coefficients of the LL subband at the current level is at least a predetermined number:

fa) assigning said predetermined number of LL coefficients as a current tile, if the number of stored LL coefficients is at least said predetermined number, and repeating steps c) to f); and

fb) repeating steps b) to f) if the number of stored LL coefficients is less than said predetermined number.

According to a third aspect of the invention there is provided an apparatus for encoding an image, said image being capable of be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specifying a block size in number of coefficients, in a first and second dimension, the apparatus comprising:

storage means for storing at least a portion of said image, the portion of the image having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

a first and second filtering means for applying the linear transform to a first and second dimension of said image portion respectively to provided a LL, LH, HL and HH subband, each subband comprising at least one coefficient;

a partial band storage means for accumulating a predetermined number of coefficients of the LL subband and using the accumulated LL coefficient as an image portion for

re-filtering by the first and second filtering means to achieve a next level decomposition;

a subband storage means for accumulating said blocks of specified size for each level in the LH, HL and HH subbands and accumulating said blocks of specified size in the LL subband at said predetermined level of decomposition;

encoder means for encoding each accumulated block into a bit stream.

Brief Description of Drawings

Notwithstanding any other forms which may fall within the scope of the present invention, preferred forms of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Fig. 1 illustrates various subbands of a four level discrete wavelet transform of an image;

Fig. 2 represents a horizontal tiling of a image as a variation on a preferred embodiment of the present invention;

Fig. 3 represents a vertical tiling of a image in accordance with the preferred embodiment of the present invention;

Fig. 4 illustrates the encoder of the preferred embodiment of the present invention;

Fig. 5 shows the tiling of Fig. 3 and a preferred tile scan order;

Fig. 6 shows the compression engine of Fig. 4 in more detail;

Fig. 7 illustrates a reorder buffer in accordance with the preferred embodiment of the present invention;

Fig. 8 represents a fill order of the reorder buffer of Fig 7;

Fig. 9 illustrates a vertical filter hardware arrangement in accordance with the preferred embodiment of the present invention;

Fig. 10 illustrates a horizontal filter hardware arrangement in accordance with the preferred embodiment of the present invention;

Fig. 11 illustrates a subband buffer hardware arrangement in accordance with the preferred embodiment of the present invention;

Fig. 12 represents a fill and scan order of the subband buffer of Fig 11;

Fig. 13 illustrates a control state machine hardware arrangement in accordance

with the preferred embodiment of the present invention;

Fig. 14 illustrates a partial band buffer hardware arrangement in accordance with the preferred embodiment of the present invention;

Fig. 15 represents a fill and scan order of the partial band buffer of Fig 14; and

Fig. 16 is a block diagram representing a functionality of the control state machine of Fig. 13.

Detailed Description of a Preferred Embodiment

Referring to Fig. 1 there is shown a discrete wavelet decomposition of an image to 4 levels. At a highest level, level 4, there is a Low-Low (LL) frequency subband (or DC subband) and a High-Low (HL4) frequency subband, a Low-High (LH4) frequency subband and a High-High (HH4) frequency subband. At a next level down there is a High-Low (HL3) frequency subband, a Low-High (LH3) frequency subband and a High-High (HH3) frequency subband for the present level. At the lowest level there is High-Low (HL1) frequency subband, a Low-High (LH1) frequency subband and a High-High (HH1) frequency subband.

Each level is generated by an application of a (single-level) discrete wavelet transform (DWT). Thus an application of the DWT to an image will generate a DC subband for a first level and three higher frequency subbands HL1, LH1 and HH1. The DWT is then applied to the DC (or LL) subband of first level to generate a DC subband for a second level and three higher frequency subbands HL2, LH2 and HH2. In turn, a further application of the DWT to the DC subband of the second level will generate a DC subband of a third level and three higher frequency subbands HL3, LH3 and HH3 and so on to a desired level of decomposition. Consequently, a four level decomposition described above requires four applications of the (single-level) DWT, a first application to the image and subsequent applications to the resultant Low-Low frequency subbands in a recursive manner as described above.

The preferred embodiment of the present invention achieves a predetermined level of decomposition, preferably for levels greater than one, in a memory efficient and low memory bandwidth (bytes per pixel) implementation. Further the implementation of the

preferred embodiment is substantially independent of a size of the image to be decomposed (for images typically greater than 1K pixel by 1K pixels). That is, a discrete wavelet decomposition of an image at a predetermined level is provided by memory buffers with total memory capacity less than that required to buffer the entire image and substantially independent of a width or height of the image.

The implementation according to the preferred embodiment of the present invention achieves efficient memory buffering and low memory bandwidth by fixing the number of maximum levels of DWT decomposition, dividing an image to be decomposed into a plurality of tiles and requiring that a first dimension of each tile be substantially a minimum number of pixels required to produced a predetermined size DC block in a corresponding (first) dimension of the DC block at the maximum level of a DWT decomposition . Further, a second dimension of each tile is less than the minimum number of pixels required to produced a corresponding (second) dimension of the predetermined size DC block. Preferably, the second dimension of each tile has a minimum length, measured in pixels, that is required to produce a (fraction) sub-multiple of a corresponding dimension of the predetermined size DC block at the maximum level.

For example, using a Daubechies 9/7 DWT filter (i.e. a 9 tap lowpass and 7 tap highpass), and entropy coding DWT blocks (arrays) of 32 coefficients x 32 coefficients (height x width), at each level to a maximum of 4 levels of DWT decomposition would require, according to the preferred embodiment, a sub-division of an image into a plurality of overlapping tiles, each tile being 617 pixel in a first dimension (e.g. height of a tile) and 71 pixels in a second dimension (e.g. width of a tile). Each tile is DWT decomposed to 4 levels, with intermediate buffering of the Low-Low frequency portions at each iteration of the decomposition, to produce a predetermined number of DC coefficients. After processing approximately nine (9) such tile, a 32 x 32 block of DC coefficient is recovered and entropy coded. Through each of the iteration, 32x32 blocks of higher frequency coefficients (eg HL, LH and HH at each level) are also entropy coded.

The preferred embodiment will be described, hereinafter, predominately with reference to a four level DWT decomposition of an image using a Daubechies 9/7 DWT filter and subsequent entropy coding of 32x32 blocks of DWT coefficients, however the invention is not limited thereto. A predetermined maximum number of level decomposition, a

different filter and/or a different block size of accumulated coefficients before entropy coding the block can be used without departing from the scope and spirit of the invention. For example a 5 level DWT decomposition can be performed and/or 64x64 bloeks of DWT coefficients can be accumulated before entropy coding a block. Further, a multitude of wavelet filters can be used including "HAAR" filter, "Le Gall 5/3", "LeGall 10/18". Such modifications may require corresponding changes in the hardware architecture described hereinafter.

Referring, now, to Fig. 2 and Fig. 3, there is shown a tiling of an image in accordance with the preferred embodiment of the present invention. That is, a sub-division of the image into a plurality of, preferably partly overlapping, tiles. The dimension of each tile being determined by fixing predetermined parameters such the level of decomposition, block size accumulated before entropy coding, type of DWT filter used. Each tile comprises 71 pixel in one dimension by 617 pixels in the other dimension of the tile. In Fig. 2 the tiles are disposed horizontally, that is, the longest side of the tile is laid out substantially parallel to scan lines of the image. Fig. 3 shows an alternate arrangement where the tiles are disposed vertically so that the shortest side is substantially parallel to the scan lines of the image. Each tile in Fig. 2 and Fig. 3 is labelled as "Tile n", where n is a positive integer number and the labelling indicates the preferred processing order of the plurality of tiles. For example, "Tile 1" is processed before "Tile 2" which is processed before "Tile 3" and so on. Further, each tile has a predetermined overlap region 200 and 201 with an adjacent tile which is determined as described below.

In general, to produce H lines of a subband at the second level of the DWT, using the Daubechies 9/7 DWT filter, $2H + 7$ lines of the LL1 subband is required, which in turn requires $2(2H + 7) + 7 = 4H + 14 + 7$ lines of an image. The seven (7) lines in addition to the $2H$ lines is required by the Daubechies 9/7 filter and is referred to as the "overlap" 200 . Preferably, the overlap is distributed as three (3) line preceding the $2H$ lines to be processed and four (4) lines after the $2H$ lines to be processed.

By mathematical induction it can be shown that, for the present filter, in one dimension to form *H lines* of any subband to level J requires,

$$2^J H + \left(2^{J-1} + .. + 2^1 + 2^0\right) \cdot 7 = 2^J H + 7\left(2^J - 1\right) \qquad \text{EQ1}$$

lines of an image. The second term in the above equation is the overlap required. Naturally, this applies to the other dimension as well. So for example, for a four level decomposition ($J = 4$) using the Daubechies 9/7 filter and accumulating blocks of 32 pixels wide (horizontal) $W = 32$, according to EQ1 would require 16x32 + 7x15 = 512+105 width tiles, where the overlap 200 is 105 pixels. That is, the overlap 200 between "Tile 1" and "Tile 11" in Fig 2 is 105 pixel in the horizontal direction by 71 pixel ($2H$+7) in the vertical.

Transforming $2H + 7$ lines at a time, and progressing to a next (overlapped) tile, in increasing label order, produces H lines from each of the HL1, LH1 and HH1 subbands. These subband portions can be coded as soon as W (in our example W=32 pixels) such pixels (by H lines) – that is for every $2H$ pixel columns input, just as was the case above. At the end of processing the first pass (i.e. one tile at one level decomposition) of a tile, H lines by $2^{J-1}W + 7\left(2^{J-1} - 1\right)$ coefficients per line of the LL1 subband needed to be buffered. This number of coefficients per line in the LL1 subband, is required, in order to form a W width block(s) at the top level (level J). Referring to Fig 2, when tile "Tile 1" is processed then "Tile 2" is processed in substantially the same manner. That is, process another $(2H + 7)$ x $\left(2^J W + 7\left(2^J - 1\right)\right)$ pixel tile of the image. Now, having processed "Tile 2" another H lines of the LL1 subband are generated, with a total of $2H$ lines after processing two tiles. To further decompose LL1 subband to produce H lines of a LL2 subband $2H$+7 lines of the LL1 subband are required. Since processing "Tile 2" has thus far produced $2H$ lines a third such tile "Tile 3" needs to be processed providing a total of $3H$ lines which provides the $2H$+7 line required to perform the next level decomposition to recover H lines of a LL2 subband. Similarly, continuing in this manner to a processing of at least $2^J H + 7\left(2^J - 1\right)$ lines will produce H lines of coefficient at the J th level of decomposition. Recall that a line is $2^{J-1}W + 7\left(2^{J-1} - 1\right)$ pixels (or DWT coefficients long) and not necessarily the width or length of the image. At each iteration to obtain a next level

decomposition the resultant coefficients of the LL frequency subband at a current level are buffered whilst each HxW block of higher frequency generated (ie LH, HL, HH) at the current level is sent to an entropy encoder for coding.

Output devices, in general, and display devices in particular display images in raster scan order, from left to right and top to bottom. Thus, the preferred implementation of the present embodiment is that of the tile arrangement shown in Fig. 3. The overlapping tiles are preferably arranged so that the shortest dimension of the tile is parallel to the scan lines of the image. Preferably, the tiles are processed across the width of the image, before a next row of tiles are processed. This arrangement is preferred so the tiles can be processed from left to right and top to bottom (ie raster scan type order), however other orientation can be used without departing from the scope and spirit of the invention. For example the tile arrangement of Fig. 2 is an alternate arrangement to that of Fig. 3.

In the arrangement of Fig. 3 each tile is of size $\left(2^J H + 7\left(2^J - 1\right)\right)$ x $(2W + 7)$ pixels and processed in raster scan order. For a $J = 4$ level DWT decomposition and to attain H=32 by W=32 coefficient blocks sizes for entropy coding, each tile of the image has dimensions 512+105 = 617 lines by 2x32+7 = 71 coefficients (or pixel) per tile line.

Referring to Fig. 4, there is shown a block diagram of the encoder 400, which takes an uncompressed image 401 and produces a compressed bit stream 402 representing the image. The encoder 400 comprises a predetermined amount of external memory 403, preferably dynamic random access memory DRAM, and a compression engine 404 which itself includes a predetermined amount of internal memory 405, typically static random access memory SRAM. The encoder 400 and in particular the compression engine 404 performs the necessary steps to produce a compressed image bit stream 402 from an uncompressed image 401 and includes transform performing a linear transform (DWT) and entropy coding the transform coefficients. Further, the compression engine 404 is preferably implemented as a single chip which uses a master pixel clock signal, **clk**, throughout the chip.

The Uncompressed image 401 is buffered through the external DRAM of the encoder 400 and is controlled by the compression engine 404. The external DRAM is preferably used to buffer a portion of the uncompressed image substantially corresponding

to a a tile as hereinbefore described (617x71 pixel or coefficients). The compression engine read short bursts of data corresponding to 71 pixels or coefficients along each line, scanning down the tile in zig-zag order.

Fig. 5 shows the zig-zag order 500 in which each tile and a more detailed illustration of the tile arrangement of Fig. 3 for a portion of an image 502. As previously described the width of a tile is $2W+7$ which for a desired block width of W=32 results in a tile width (or the length of a line in a tile) of 71 pixels (ie. 64 + 7, seven being the horizontal overlap 201). Thus the compression engine 404 preferably reads one line of a tile at a time working in zig-zag order until 617 lines or $\left(2^J H + 7(2^J - 1)\right)$, where H=32, lines are read. That is, the $7(2^J - 1)$ portion of the previous expression constitutes a vertical dimension of an overlap region 200 between adjacent tiles. For example, for a four level DWT the overlap region 200 is 105 pixels in length by the width (71 pixels) of a tile. As each 32x32 (HxW) transformed block of higher frequency components (LH, HL, HH) is generated the block is entropy code without intermediate buffering. Thus in the preferred embodiment the overlap region 201 between two adjacent tiles (eg. "tile1" and "tile 2") in a horizontal direction is substantially seven (7) pixels wide by the length of a tile and the overlap region 200 two adjacent tiles (ie. "tile 1" and "tile i") in a vertical direction is substantially one hundred and five (105) pixels long by the width of a tile

Referring now to Fig. 6 there is shown the compression engine 404 of Fig 4 in more detail. The compression engine 404 comprises: a reorder buffer module 601 which receives 602 data from the external DRAM 403 in response to a memory address transmitted to the DRAM via an address bus 603; a control state machine module 604; a vertical filter 605; a horizontal filter module 606; a subband buffer 607, a partial band buffer 608; and an entropy encoder unit 609;

The Reorder Buffer module 601 controls the reading and writing to the external DRAM 403 and also includes a small memory buffer to assemble the data received from the DRAM 403 so that it can be read out to the vertical filter 605 in a predetermined order. The reorder buffer module 601 passes nine bytes (72 bits) of data, at a time, to the vertical filter module 605. The vertical filter module 605 performs a low pass and high pass filtering in a first dimension (the vertical dimension of the tile of Fig. 3). There are

two outputs, 611 and 610, from the vertical filter module 605, corresponding to low pass and high pass filtering of the data. Each output 610 and 611 passes to separate horizontal filters in the Horizontal Filter module 606 where an identical filtering operation is performed in a second dimension (the horizontal dimension of the tile of Fig.3). The Horizontal Filter module 606 provides four output signals corresponding to a LL, LH, HL and HH subband signal. The subband signals comprising high frequency components (ie LH, HL and HH) are temporarily buffered in the Sub-Band Buffer module 607 before being encoding by the entropy encoder module 609 to produce an encoded bit-stream output on 613. The LL subband signal is passed to the Partial Band Buffer 608 where it is assembled into larger blocks which are then passed back into the vertical filter module 605 for subsequent transform levels. Only at a final level (eg level 4) is the LL subband signal encoded into the output bit stream (via output 613) . The Control State Machine module 604 oversees operation of the compression engine 404, making sure that appropriate data are stored in the correct places, and that operations are done on the correct data at the right times.

Turning to Fig. 7, where the Reorder Buffer Module 601 of Fig. 6 is shown in more detail. The Reorder Buffer Module 601 includes random access memory (hereinafter Reorder RAM) 700 to store intermediate raw data received 602 from external memory 403 and a Reorder Address Generator sub-module 701 which includes a DRAM controller for controlling the external DRAM 403. A `data_valid` 702 is a signal received from an external device supplying (raw) pixel data 401 to the DRAM 403 indicating that the current data on the DRAM data pins is valid. A `data_ack` signal 703 acknowledges that the data has been read into the DRAM 403 . The external DRAM 403 takes its input directly from the external raw pixel supply (not shown). The Reorder Address Generator 701 determines and supplies 704 addresses to the Reorder RAM 700 as well as controlling writes into that RAM.

The Data Out (DO) pins from the Reorder RAM 700 are connected to the Data In (DI) pins of the same Reorder RAM 700 with a shift of 8 bits. That is the DO(n+7:n) pin is connected to the DI(n+15:n+8) pin, where n represents the lower bit number. For example, DO(7:0) pin representing eight least significant bits of a 72 bit word Data Out is connected to DI(15:8) pin corresponding to a second eight least significant bits of a 72 bit word Data In. The data is written into the reorder RAM 700 using a read-modify-write scheme. An

eight least significant bits of a 72 bit word Data In, that is , the byte input to the DI(7:0) pin comes from external DRAM 403 while the rest of the input data word comes from the Reorder RAM 700 outputs in the shifted fashion described above. The Reorder RAM thus acts like a big byte-wide addressable shift register, 9 words deep and 71 words wide. Data is read from the external DRAM in bursts of 71 words along a scan line of the image (ie a line of a tile). Two lines (71 bytes x 2) of image data are written into the Reorder RAM 700 for each row (lines) of samples that are used by the Vertical Filter Module 605. This constitutes or provides a down sampling (decimation by two) of the image data in the vertical direction as required by the DWT.

As previously described the re-order data from the Reorder RAM 700 is passed, via a ro_data bus 705, on to the Vertical Filter Module 605 nine (9) bytes at a time (72 bits).

Fig. 8 is a representation of the organisation of the Reorder Buffer 601 of Fig. 7. The Reorder Buffer 601 of the preferred embodiment is 71 pixels wide and 9 lines high. The buffer 601 is written from left to right as indicated by the arrow 801 in Fig 7. As each line is written from left to right, pixels are pushed down the lines, one at a time. That is, when nine lines of the buffer have been written, (9x71=) 639 pixels will occupy the buffer and a first pixel of the 639 pixels to enter the buffer resides at cell 802 of the buffer at a first position of the ninth line 804 of the buffer, whilst a last pixel of the 639 pixels to enter the buffer resides at cell 803 of the buffer on the first line 805. The memory 700 (Reorder RAM) is organised as 71 words of 72 bits where the 72 bits represent 9 pixels in the vertical dimension. After 9 lines have been written to the same 71 addresses repeatedly (8 bits at a time), the entire buffer 601 is full. As a last 71 bytes (pixels), of the 639 pixels, is written into the top line, the data is read out during the read-modify-write cycles (all 72 bits) is used to load the Vertical Filter Module 605. Once the buffer 601 has been filled, data is used by the Vertical Filter Module 605 from every second line of image data loaded into the Reorder Buffer 601, which in effect performs a sub-sampling in a vertical direction.

The sub-sampling in a vertical direction is achieved as follows. Initially, the Vertical Filter Module 605 processes nine lines of the reorder buffer with a first tap of the 9 tap filter arranged to use data from the last line 805 of the reorder buffer while a last tap of the filter uses the data from the first line 804, of the reorder buffer 601, to produce, for

example, a line of low pass filtered data which may be stored in the partial band buffer 608. Naturally, taps intermediate the first and last tap use data from corresponding lines of the reorder buffer 601. Next, each line of data the reorder buffer 601 is "pushed-down" the buffer so that the last line 804 of the reorder buffer 601 is in effect deleted (pushed off the buffer) and the data from each preceding line is written into a subsequent line of the reorder buffer 601. That is, data from the 8th line 806 of the buffer is written into the 9th line 804, data from the 7th line of the buffer is written into the 8th line 806, data from the 6th line is written into the 7th line and so on. On the first line 805 of the reorder buffer 601 a new line of data is inputted from external DRAM 403. This push done is repeated for every line inputted into the reorder buffer 601. Thus sub-sampling in the vertical direction is performed by reading out and processing nine lines of the reorder buffer after every second line of data is inputted into the reorder buffer 601. The reorder buffer is completely refilled at the start of a tile and the above described process is repeated preferably until the entire image is processed.

After substantially 71 lines of the image have been retrieved from the external memory 403 and transferred to the Reorder Buffer 601, the Sub-Band Buffer 607 contains sufficient transform coefficients for encoding to commence. In the implementation according to the preferred embodiment the Sub-Band Buffer 607 becomes full and the coefficients need to be encoded before the Reorder Buffer 601 can be further updated. While higher level coefficients are being filtered, new data can be loaded into the Reorder Buffer in preparation for the next level one filter operation.

The Reorder Address Generator 701 simply cycles from 0 to 70 for the Reorder Buffer 601, but addresses the external DRAM 403 in zig-zag order, as hereinbefore described with reference to Fig. 5, during reads. The Reorder Address Generator 701 also controls the raster scan order during writes of the image into the external DRAM 403. The external DRAM 403 is preferably large enough to store at least 617 full-width lines of image. In practice, good bandwidth can be realised from a dual port DRAM, or optionally a Video Random Access Memory (VRAM) can be used for the external memory 403.

Fig. 9 represents the Vertical Filter Module 605 of Fig. 6 in more detail. The Vertical Filter Module 605 includes a multiplexer 901 to select either data from the Reorder Buffer 601 when doing a level one decomposition or data from the Partial Band Buffer

608 during subsequent decomposition levels. The 9 bytes x 8 bits of data received from the Reorder Buffer 601 is converted into 9 words of 16 bits to be compatible with the data from the Partial Band Buffer 608 by zeroing the unused bits. The multiplexer 901 is controlled by the control state machine module 604. Typically, eight bit colour channel per pixel data is converted to sixteen bit precision DWT coefficient so that a minimal amount of information is lost due to rounding effect on the coefficient values.

In addition the Vertical Filter Module 605 includes a Vertical Low Pass Filter 902 and a Vertical High Pass Filter 903 which receive data in accordance with the selection of the multiplexer 901. The Vertical Low Pass Filter 902 takes 9 x 16 bit samples and performs a matrix operation (multiply-accumulate) on the data, producing a 16-bit result (**vfl_data**) from every 9 words of input. The Vertical High Pass Filter 903 only requires 7 x 16 bit samples to perform its matrix operation, again producing 16 bits of result (**vfh_data**). As previously described the filters of the present embodiment are based on the Daubechies 9/7 DWT, however other wavelet filters can be used.

Fig. 10 is the Horizontal Filter Module 606, of Fig. 6, in more detail. The Horizontal Filter Module 606 comprises two substantially identical sections 1004 and 1005 which process data from the Vertical Filter Module 605. Each section comprises a 9-Stage Shift Register 1001, a Horizontal Low Pass Filter 1002 and a Horizontal High Pass Filter 1003. Data first passes through a pair of 9-Stage Shift Registers which buffer up to nine (9) consecutive samples received from the Vertical Filter Module 605 to be processed simultaneously. Whilst the two sections 1004 and 1005 process the data in substantially the same manner, the source of the data for each section is different. One section 1004 receives the vertical low pass filter signal **vfl_data**, while the other section 1005 receives the vertical high pass filter signal **vfh_data**.

The Horizontal Low Pass Filter 1002 of each section take 9 x 16 bit samples and perform a matrix operation (multiply-accumulate) on the data, producing a 16-bit result each from every 9 words of input, producing a **ll_data** signal 1006 and a **hl_data** signal 1007 for section 1004 and 1005 respectively . The Horizontal High Pass Filter 1003 of each section only requires 7 x 16 bit samples to perform a matrix operation, again each High Pass Filter 1003 producing 16 bit result. The Horizontal High Pass Filter for section 1004 provides signal **lh_data** 1008 and the Horizontal High Pass Filter for section 1005

provides signal **hh_data** 1009. The resultant subband data, that is signals **ll_data**, **lh_data, hl_data** and **hh_data** corresponding to coefficients of the LL, LH, HL and HH subbands are passed on to the Sub-Band Buffer module 607.

After processing 9 word samples both 9-stage shift registers shift the data by two to account for sub-sampling (decimation by two) in the horizontal direction.

The **ll_data** 1006, corresponding to coefficients of the LL subband at the current level of DWT decomposition, is also sent to the Partial Band Buffer module 608 for subsequent levels of decomposition.

The Sub-Band Buffer Module 607 of Fig 6 is shown in greater detail in Fig. 11. The Sub-Band Buffer 607 includes: a predetermined amount of random access memory (RAM) 1101 to store a predetermined number of coefficients from each of the LL, LH, HL and HH subbands; a Sub-Band Buffer Address Generator sub-module 1102 and a multiplexer 1103. The Sub-Band Buffer Address Generator 1102 supplies addresses to the Sub-Band RAM 1101 as well as controlling writes into that RAM under the control of the Control State Machine module 604.

The multiplexer 1103 selects data, coefficients, from one of the four subbands for encoding by the Entropy Encoder Unit 609.

Referring to Fig. 12 there is shown a representation of a data arrangement or organisation of the Sub-Band Buffer of Fig. 11. The Subband Buffer holds four (4) banks 1201 of thirty two (32) by thirty two (32) coefficients and each coefficient is represented by a sixteen (16) bit value. Thus, the Sub-Band RAM is a total of 64 Kbits and is organised as 1024 words of 64 bits, storing one coefficient from each subband at each memory location in parallel. That is, four coefficients, one from each subband at a current level, are represented each as sixteen (16) bits of a sixty four (64) bit word. The buffer 1101, and in particular each bank 1201, is filled in raster scan order from top to bottom as indicated by the zig-zag pattern 1202 shown in Fig. 12. During encoding at the Entropy Encoding Unit 609, the Subband RAM 1101 is read out in quadtree manner.

Referring to Fig. 13 there is shown the Entropy Encoder unit 609 of Fig. 6 in more detail. The Entropy Encoder unit 609 comprises a Bitplane Converter module 1301 and Bitplane Buffer sub-module 1302 comprising sixteen (16) bitplane memories; a plurality of Bitplane Tree Builder units1303 and corresponding Bitplane Tree buffer units 1304; a

Bitplane Encoder 1305; and a plurality of intermediate buffers 1306 for intermediate storage of data use by the Bitplane Encoder 1305. The data stored in Sub-band RAM 1101 is read in quadtree order from the Subband Buffer module 607. The bitplane converter module 1301 splits up each bit plane's data and writes into each of 16 bitplane memories in the Bitplane Buffer submodule 1302. The data is also scanned by the plurality Bitplane Tree Builders which write processed bit plane tree structure information into corresponding Bitplane Tree buffers 1304. Data from each Bitplane Tree 1304 and the Bitplane Buffer 1302 is used by the Bitplane Encoder 1305 to perform a bit-stream generation which is output (via bus 613). The Bitplane Encoder 1305 is synchronised (controlled) by the control state machine module 604. The intermediate buffers 1306 comprise: a buffer for temporarily storing a List of Significant Coefficients (LSC); a buffer for temporarily storing a List of Insignificant Coefficients (LIC) ; and a buffer for temporarily storing an Old Insignificant Region (OIR) list which can also be implemented as two buffers as a first list of regions buffer and a second list of region buffer (not shown in Fig. 13). In addition the Entropy Encoder Unit 609 includes a sign bit buffer (not shown in Fig. 13) which is connected to the output of the Bitplane Converter module 1301 and to an input to the Bitplane Encoder 1305 which also encodes a sign bit into the bit-stream. An example of an Entropy Encoder unit is described in more detail in Appendix A appended hereto.

The preferred embodiment of the present invention is described with reference to an Entropy Encoding Unit which performs a quadtree bitplane encoding as described above, however other entropy encoders can be used without departing from the scope or spirit of the invention. For example, a bitplane Arithmetic coder can be used to encode the transform coefficients, or optionally a hybrid encoder performing part quadtree encoding and part Arithmetic coding of the DWT coefficients.

Referring to Fig 14 there is illustrated in more detail the Partial Band Buffer module of Fig 6. The Partial Band Buffer Module 608 includes: RAM 1401 to store the LL subband data (i.e. **ll_data** 1006 generated by the Horizontal Filter Module 606 of Fig.6) for each level of decomposition ; a Partial Band Buffer Address Generator submodule 1402 and a barrel shift register (barrel shifter) 1403. The Partial Band Buffer Address Generator 1402 supplies addresses to the Partial Band RAM 1401 as well as controlling writes into that RAM under the control of the Control State Machine module 604.

The Partial Band Buffer holds LL subband data from each level of compression until enough samples are available to filter to a next level of decomposition. Once enough samples are available at any one level, the data is read out through the barrel shifter 1403 into the Vertical Filter module 605 to filter to a next level of decomposition. Conceptually, the RAM 1401 comprises three separate buffers, each one being one hundred (100) cells wide with each cell capable of storing a 16 bit value. A first buffer, of the three separate buffers, is 305 rows (lines) deep, a second buffer is 149 rows deep and a third buffer is 71 rows deep to a total of 525 rows (lines). However, since Vertical Filter module 605 takes nine (9) values as required by the Daubechies 9/7 filter, it is desirable that the total number of rows (lines) is a multiple of nine (9). Thus, an extra 6 rows of cells are appended to the buffers resulting in a total of 531 rows (ie 59 multiple of 9). For the present embodiment this result in a memory size of at least 59 x 100 x 144 = 531 x 100 x 16 bits or approximately 0.81 Megabits.

The Barrel Shift register 1403 allows single clock alignment of data to the desired row required by the Vertical Filter module 605. Predetermined memory locations need to be read from the Partial Band RAM 1401 to produce a single word for the Vertical Filter 605. If the required single word wraps across a 9-line boundary, two adjacent memory location need to be read. This will typically occur in 8 out of every 9 line sets (each set comprising nine lines). In this case data is latched inside the Barrel Shifter 1403 until both words are read and all required bits are available and properly aligned.

Turning to Fig. 15 there is shown a representation of the data organisation of the partial band buffer 608. The Partial Band Buffer 608 is 100 coefficients wide and 525 coefficients high, plus an extra 6 of 100 coefficient wide rows (not shown in Fig 15) as described above. The Partial band buffer 608 is written to with data (coefficients) after at least one level DWT decomposition of a tile is performed. Thus a tile of

$\left(2^J H + 7\left(2^J - 1\right)\right)$ x $(2W+7)$ of the image reduces to $(2^{j-1}H + 7(2^{j-1} - 1)) \times W$

tile of coefficient after one level of decomposition. That is a 617 x 71 pixel tile after a first level of decomposition reduces to a tile of 305x32 coefficients. The Partial band buffer 608 of the preferred embodiment, at 100 coefficients wide, is thus designed to accommodate at least width-wise three 32-coefficient wide tiles 1501 and a 4-coefficient wide block

1502. Each 32-coefficient wide tile is written in scan line order left to right and top to bottom, one level at a time as shown in Fig. 15. That is, the three 32-coefficient wide tiles 1501 of coefficients for second level decomposition (each of length 256+49=305 coefficients) are fill each in zig-zag order. The 4-coefficient wide block 1502 , at least for a first tile process, is filled with a mirror image of the first four columns of coefficients of a first 32-coefficient wide tile 1501 (hereinafter referred to as "padding" or "padded"). Padding is usually performed to account for boundaries of an image where pixel data outside the image boundary is non-existent. Those skilled in the art will recognise that other padding techniques are available to account for an image boundary. For example a simplest, but not necessarily the best, option is to pad the boundary of an image with zeros (0).

As the coefficients for the second level decomposition are processed through the Vertical filter 605 and Horizontal filter 606 modules, coefficients for a third level decomposition are accumulated in a next level 32-coefficient wide tile 1504 and (128+21) = 149 coefficients in length. Again, initially a 4-coefficient wide block 1503 is padded with coefficients of a corresponding (adjacent) tile 1504 . As the coefficients of the third level decomposition are processed through the Vertical filter 605 and Horizontal filter 606 modules, coefficients for a four level decomposition are accumulated in a next level 32-coefficient wide tile 1506. Again, initially a 4-coefficient wide block 1505 is padded with coefficients of a corresponding (adjacent) tile 1506. In cases other than when at a boundary of an image were the 4-coefficient wide blocks 1502, 1503 and 1505 need to be padded the 4-coefficient wide blocks contain a last 4 columns of a previous tile, at that level, processed.

The 4-coefficient wide block at each level is used as overlap for filtering the horizontal lines. In order to generate one complete 32-coefficient wide tile at one level, requires 71 coefficients width at a previous level. The shaded area 1507 of Fig. 15 consisting of 305 x 71 coefficients for second level decomposition is required to generate a tile of 149 x 32 coefficients for third level decomposition. The coefficients in the shaded area 1507 are read out in scan line order, 71 to the line from left to right and top to bottom. Once the Partial Band buffer 608 is filled to a desired amount (eg 71 columns) at a next (higher) level, column addresses at a current level are incremented by sixty four (64) columns to the right, wrapping back to the first column as the addresses exceed the hundredth

column. That is, for example processing of a next seventy one (71) columns of coefficients, at a current level, is commenced at the sixty fourth column 1508 of the buffer 608 and wrapping around to the first column of the buffer 608. Column sixty four (64) to column sixty eight (68), now provides a 4-coefficient wide block 1509 overlap necessary to process another seventy one (71) columns of data.

As each level of the Partial Band Buffer 608 is filtered, results of the filter (LL subband coefficients) are written to a next higher level of the Partial band buffer 608 shown in Fig 15. When a highest level has been filtered (eg "level 4" of Fig 15) a resulting LL subband coefficients are written to the Sub-Band Buffer 607 without further writing these coefficients into the Partial Band Buffer 608 and are encoded straight away, without further filtering.

When reading out of the Partial Band Buffer, 9 x 16 = 144 bits are read at a time. These 9 words represent 9 vertical coefficients from a single column of the buffer. Because the line address is incremented by two between reading complete lines out of this buffer, it is necessary to pass the 144-bit data stream through a barrel shifter to align the data with the data bus into the Vertical Filter. It is also necessary to read 2 words in succession to get one 144-bit word, building up the two parts of the word by holding the required bits of the first word in registers while the required bits from the second access are fetched. Only when the alignment of the line address is a multiple of 9 will one fetch only be required. In order to be able to fetch 144-bits at a time, the memory must physically be 144 bits wide. Extra decoding is required (so that just 16 bits can be written at a time) in the form of word write enables to the memory array.

Referring to Fig 16, there is illustrated a flow diagram 1600 for the Control State Machine module 604 of Fig. 6. The Control State Machine Module 604 is responsible for controlling the function of each module 601, 604-609 and for correctly routing data to each module in synchronism with a modules function (ie at the right time).

At commencement 1601 of image compression a "**level**" variable is set 1602 to one (1), which typically represents a first level of decomposition to be performed. A decision block 1603 is entered to determine if the current level of DWT decomposition is the first level. If decision block 1603 returns true (yes) then a first nine (9) lines of the image stored in DRAM 403 (Fig. 4) are filtered 1604 to produce a first line of sub-band data.

Input data is retrieved from the external DRAM 403 for level 1. Otherwise the decision block 1603 returns false (no) causing the flow to proceed to block 1605 where data is retrieved from the internal Partial Band Buffer 608 and filtered (DWT decomposed to a next level). Enough data is retrieved from the Partial Band Buffer 608 to produce a line, of a tile, of DWT coefficients of a next level. These results are stored in the Partial Band Buffer 608 of a location for a corresponding level as previously described.

After 71 lines of data are filtered, 32 lines of subband buffer 607 storage are filled. At this point in the flow diagram 1600 a check 1606 is made to see if the subband buffer 607 is full. If this check 1606 returns false (no) the process flow is returned to decision block 1603 and continues as described above with reference to decision block 1603. If check block 1603 returns true (yes) a further decision (or check) block 1607 is entered and a check is made to see if the data is within an overlap region 200 for a level that is being filtered. If not, that is block 1607 returns false (no) then three subbands LH, HL and HH stored in the subband buffer 607 are encoded 1608. Following the encoding of the three subbands a decision block 1609 is entered and a check is made to determine if a current level of decomposition is a maximum level (that is level 4). If the current level is 4 decision block 1609 returns true (yes) a LL subband in the subband buffer 607 is also encoded 1610 and the flow process continues at decision block 1611. Further, at decision block 1609 if false (no) is returned the flow process continues at decision block 1611 without encoding the LL subband in the subband buffer 607.

Another entry to decision block 1611 is via decision block 1607, where decision block 1607 returns true (yes). That is, the data being processed is within an overlap region 201 or 200 of adjacent tiles. The check performed at decision block 1607 is used to prevent the encoding of the overlap region twice. The process flow proceeds to decision block 1611 when decision block 1607 returns true (yes).

At decision block 1611 a check is then made to see if at level 2 of the partial band buffer 608 (Fig. 6) has a complete set of 71 columns of (256 + 105) rows of coefficients. If so, then the **level** variable is set 1612 to two (2) and processing is resumed at decision block 1603 and continues through steps 1605 to 1611, as described above, with data retrieved from level two (2) of the partial band buffer 608.

If the level two (2) partial band buffer is not ready to be filtered, that is, a complete

set (ie. a tile at level two (2) comprising 71 by 361 coefficients) is not present in the partial band buffer 608, then false is returned by decision block 1611 and check (decision) block 1613 is entered. At check block 1613 a check is made on whether at level three (3) of partial band buffer a complete set of 71 columns of (128 + 21) rows of coefficients are present. If so, then the **level** variable is set 1614 to three (3) and processing is resumed at decision block 1603 and continues through steps 1605 to 1613, as described above, with data retrieved from level three (3) of the partial band buffer 608.

If the level three (3) partial band buffer 608 is not ready to be filtered, that is, a complete set (ie. a tile at level three (3) comprising 71 by 149 coefficients) is not present in the partial band buffer 608, then false (no) is returned by decision block 1613 and check (decision) block 1615 is entered. At check block 1615 a check is made on whether at level four (4) of partial band buffer a complete set of 71 columns of (64 + 7) rows of coefficients are present. If so, then the **level** variable is set 1616 to four (4) and processing is resumed at decision block 1603 and continues through steps 1605 to 1615, as described above, with data retrieved from level four (4) of the partial band buffer 608.

If the level four (4) partial band buffer 608 is not ready to be filtered, then false (no) is returned by decision block 1615 and check (decision) block 1617 is entered. At block 1617 a check is made to see if the entire image has been completely processed. If this is the case decision block 1617 returns true (yes) and the flow process terminates 1618. Otherwise, decision block 1617 returns false (no) and processing is resumed at block 1602, assigning the **level** variable one (1) and the processing is continued and described above with reference to steps 1602 to 1617 again with data retrieved from the external DRAM 403.

In the embodiment described, nine (9) source lines are required to generate the first line of coefficients, whereas two (2) source lines are needed to generate a next and each subsequent line of coefficients. A minimum of 71 lines of 71 pixels (or coefficients) are required to make 32 lines of 32 coefficients. The partial band buffer 608 is initially loaded with three tiles of coefficients in order to process a next level. For every subsequent tile to be generated at the next level only two additional tiles of input are necessary since the partial band buffer 608 already contains at least one tile from previously load data. In the present embodiment requires 71 columns of coefficients, in the partial band buffer, for a

current level to allow filtering to a next level. That is, filtering starts with four (4) columns of coefficient from a previously processed tile and uses two additional (complete) tiles (32 columns each) and three (3) columns from a next tile to be processed to make up a 71 column tile. Optionally, the Partial Band Buffer 608 can be implemented as three separated memory units one for each level of decomposition rather than implementing the buffer 608 as a single buffer memory as illustrated in Fig. 15. This option may require separate read and write address generators.

The foregoing describes a preferred embodiment of the present invention, however, modifications and/or changes can be made thereto by a person or persons skilled in the art without departing from the scope and spirit of the invention. The present embodiment is, therefore, to be considered in all respects to be illustrative and not restrictive.

Aspects of the Invention

The following numbered paragraphs set forth aspects of the invention, including:

1 A method of encoding an digital image comprising a plurality of pixels, said image being able to be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specified block size in number of coefficients, in a first and second dimension, the method comprising the steps of:

a) dividing the image into a plurality of tiles, each tile having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

b) selecting a current tile;

c) decomposing said current tile using the DWT to at least one level of decomposition to form a plurality of subbands including a LL, LH, HL and HH subband;

d) accumulating coefficients in each subband of the LH, HL and HH subbands to form blocks of said specified size and encoding each said block to a bit stream;

e) accumulating LL subband coefficients and repeating steps b) to e) until a predetermined number of coefficients of the LL subband have been accumulated;

f) assigning as a current tile said predetermined number of accumulated LL subband coefficients;

g) repeating steps c) to g) until the predetermined level of decomposition is reached; and

h) encoding the LL subband into the bit stream.

2 A method as set out in paragraph 1, wherein accumulating coefficients in each subband includes accumulating coefficients in corresponding subbands of different tiles.

3 A method as set out in paragraph 1 or 2, wherein encoding comprises encoding each block substantially as each block is accumulated.

4 A method as set out in any one of the preceding paragraphs, wherein said specifying a block size comprises H coefficients in said first dimension by W coefficients in said second dimension.

5 A method as set out in paragraph 4, wherein the number of coefficients in the first dimension is equal to the number of coefficients on the second dimension (W=H)

6 A method as set out in paragraph 4 or 5, wherein said predetermined level is J and the minimum number of pixels in a first dimension of each tile is defined by $2^{J}H + O(2^{J} - 1)$ where O is an overlap required by the DWT filter.

7.A method as set out in paragraph 6, wherein the number of pixels in a second dimension of each tile is less than that of the first dimension and is defined by $2W$+ O.

8 A method as set out in any one of the preceding paragraphs, wherein said DWT filter is a Daubechies 9/7 filter and the overlap required is 7 pixels or coefficients (O=7).

9 A method as set out in any one of the preceding paragraphs, wherein said DWT filter is a Haar filter and the overlap required is no pixels or coefficients (O=0).

10 A method of encoding an digital image comprising a plurality of pixels, said image being able to be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specifying a block size in number of coefficients, in a first and second dimension, the method comprising the steps of:

a) dividing the image into a plurality of tiles, each tile having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

b) selecting a tile of the image as a current tile;

c) decomposing said current tile using the DWT filter to a provide a plurality of coefficients in a LL, LH, HL and HH subbands;

d) encoding coefficients of the LH, HL, HH subband of a current level into a bit-

stream;

e) determining if said current level is the predetermined level:

ea) encoding the coefficients of the LL subband into the bitstream, if the current level is the predetermined level, and repeating steps b) to e); and

eb) storing coefficients not previously encoded of the LL subband if the current level is not the predetermined level;

f) determine if the number of said stored coefficients of the LL subband at the current level is at least a predetermined number:

fa) assigning said predetermined number of LL coefficients as a current tile, if the number of stored LL coefficients is at least said predetermined number, and repeating steps c) to f); and

fb) repeating steps b) to f) if the number of stored LL coefficients is less than said predetermined number.

11 A method as set out in paragraph 10, wherein said specified block size comprises H coefficients in said first dimension by W coefficients in said second dimension.

12 A method as set out in paragraph 11, wherein the number of coefficients in the first dimension is equal to the number of coefficients on the second dimension of said specified block (W=H)

13 A method as set out in paragraph 11 or 12, wherein said predetermined level is J and the minimum number of pixels in a first dimension of each tile is defined by $2^{J}H + O(2^{J} - 1)$, where O is an overlap required by the DWT filter.

14.A method as set out in paragraph 13, wherein the number of pixels in a second dimension of each tile is less than that of the first dimension and is defined by $2W$+ O.

15 A method as set out in any one of paragraphs 10 to 14, wherein said DWT filter is a Daubechies 9/7 filter and the overlap required is 7 pixels or coefficients (O=7).

16 A method as set out in any one of paragraphs 10 to 14, wherein said DWT filter is a Haar filter and the overlap required is no pixels or coefficients (O=0).

17 A method as set out in any one of the preceding paragraphs, wherein said encoding is preformed by a bit-plane entropy encoder.

18 A method as set out in any one of the preceding paragraphs, wherein said encoding is preformed by an Arithmetic Encoder.

19 A method as set out in any one of the preceding paragraphs, wherein said encoding is preformed by a hybrid encoder comprising an Arithmetic and Bit-plane entropy Encoder.

20 An apparatus for encoding an image, said image being capable of be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specifying a block size in number of coefficients, in a first and second dimension, the apparatus comprising:

a) means for dividing the image into a plurality of tiles, each tile having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

b) selecting means for selecting a current tile;

c) decomposing means for decomposing said current tile using the DWT to at least one level of decomposition to form a plurality of subbands including a LL, LH, HL and HH subband;

e) means for accumulating predetermined number of coefficients of the LL subband coefficients;

f) means for assigning as a current tile said predetermined number of accumulated LL subband coefficients;

g) means for feeding back the current tile to the decomposing means for decompositing the current tile to a further levels of decomposition;

d) means for accumulating coefficients in each subband of the LH, HL and HH subbands to form blocks of said specified size and accumulating at least one block of said specified size in the LL subband at said predetermined level; and

e) encoding means for encoding each said block to a bit stream;

21 An apparatus for encoding an image, said image being capable of be transformed by a discrete wavelet transform (DWT) to a predetermined level of decomposition and capable of being encoded on a block by block basis, each block having a specifying a block size in number of coefficients, in a first and second dimension, the apparatus comprising:

storage means for storing at least a portion of said image, the portion of the image having substantially a minimum number of pixels required to produce the number of coefficients in the first dimension of said block at said predetermined level of DWT decomposition by less than a minimum number of pixels required to produce the number of coefficients in the second dimension of said block at said predetermined level of DWT decomposition;

a first and second filtering means for applying the linear transform to a first and second dimension of said image portion respectively to provided a LL, LH, HL and HH subband, each subband comprising at least one coefficient;

a partial band storage means for accumulating a predetermined number of coefficients of the LL subband and using the accumulated LL coefficient as an image portion for re-filtering by the first and second filtering means to achieve a next level decomposition;

a subband storage means for accumulating said blocks of specified size for each level in the LH, HL and HH subbands and accumulating said blocks of specified size in the LL subband at said predetermined level of decomposition;

encoder means for encoding each accumulated block into a bit stream.

22 An apparatus as set out in paragraphs 20 or 21, wherein said encoder means is a bit-plane entropy encoder.

23 An apparatus as set out in paragraphs 20 or 21, wherein said encoder means is an Arithmetic Encoder.

24 An apparatus as set out in paragraphs 20 or 21, wherein said encoder means is a hybrid encoder comprising an Arithmetic Encoder and Bit-plane entropy Encoder.

Dated 12 March, 1999
Canon Kabushiki Kaisha
Patent Attorneys for the Applicant/Nominated Person
SPRUSON & FERGUSON

APPENDIX A

AN ENCODER FOR REPRESENTING A DIGITAL IMAGE

Field of Invention

The present invention relates to an encoder for generating a coded representation of a digital image. The invention also relates to a method for generating a coded representation of a digital image. Furthermore, the invention also relates to a computer program product including a computer readable medium having recorded thereon a computer program for generating a coded representation of a digital image.

Background of Invention

The field of digital data compression and in particular digital image compression has attracted great interest for some time.

In the field of digital image compression, many different techniques have been utilized. In particular, one popular technique is the JPEG standard, which utilizes the discrete cosine transform to transform standard size blocks of an image into corresponding cosine components. The JPEG standard also provides for the subsequent compression of the transformed coefficients.

Recently, the field of wavelet transforms has gained great attention as an alternative form of data compression. The wavelet transform has been found to be highly suitable in representing data having discontinuities such as sharp edges. Such discontinuities are often present in image data or the like.

Although the preferred embodiments of the present invention will be described with reference to the compression of image data, it will be readily evident that the preferred embodiment is not limited thereto. For examples of the many different applications of Wavelet analysis to signals, reference is made to a survey article entitled "Wavelet Analysis" by Bruce et. al. appearing in IEEE Spectrum, October 1996 pages 26 - 35. For a discussion of the different applications of wavelets in computer graphics, reference is made to "Wavelets for Computer Graphics", I. Stollinitz et. al. published 1996 by Morgan Kaufmann Publishers, Inc.

It would be desirable to provide a method and hardware embodiment of an encoder and method so as to provide for efficient and effective encoding of a series of coefficients in order to substantially increase the speed of encoding.

Aspects of Invention

It is an object of the present invention to ameliorate one or more disadvantages of the prior art.

One or more exemplary aspects of the invention are listed below, but are not limited thereto.

According to one aspect of the invention there is provided an encoder for generating a coded representation of a digital image, said encoder including:
an input means for inputting a block of coefficients of said digital image; a plurality of tree builders, wherein each tree builder generates a tree and nodes based on a corresponding bitplane of said block of coefficients, and each said node corresponds to one of a plurality of regions of said coefficients or to one of said coefficients and each said node having a data value indicative of the significance of said one region or said one coefficent for that bitplane; a bitplane converter for converting the block of coefficients into their respective bitplanes; and a bitplane encoder coupled to said plurality of tree builders and said bitplane converter, wherein said bitplane encoder using said trees and bitplanes outputs a coded representation of the bitplanes in sequence to produce a coded representation of the digital image.

According to another aspect of the invention there is provided a method of generating a coded representation of a digital image.

According to still another aspect of the invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for generating a coded representation of a digital image.

Brief Description of the Drawings

Embodiments of the invention are described with reference to the drawings, in which:

Fig. 1A illustrates an original image;

Fig. 1B illustrates a DWT transformation of the original image of Fig. 1;

Fig. 2 illustrates a second level DWT transformation of the original image of Fig. 1;

Fig. 3 illustrates a four level DWT transformation of the original of Fig. 1;

Fig. 4 illustrates the tiling of the subbands into 32x32 blocks;

Fig. 5 illustrates an encoder in accordance with a preferred embodiment of the invention;

Fig. 6 illustrates a portion of a tree constructed by a bitplane tree builder of Fig. 5;

Fig 7 illustrates a general-purpose computer for implementing the proposed method;

Detailed Description

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features

have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

Preferred Embodiment(s) of Method

The applicant proposes herein a new method for producing a coded representation of coefficients of a transform of an image. In particular, they propose a new method for embedded block coding using quadtrees. The invention is directed to a preferred apparatus for implementing such a method.

The proposed method proceeds initially by means of a wavelet transform of image data. An overview of the wavelet process will now be described with reference to the accompanying drawings.

Referring initially to Figs. 1A and 1B, an original image 1 is transformed utilizing a Discrete Wavelet Transform (DWT) into four sub-images 3-6. The sub-images or subbands are normally denoted LL1, HL1, LH1 and HH1. The one suffix on the subband names indicates level 1. The LL1 subband is a low pass decimated version of the original image.

The wavelet transform utilized can vary and can include, for example, Haar basis functions, Daubechies basis functions etc. The LL1 subband is then in turn utilized and a second Discrete Wavelet Transform is applied as shown in Figure 2 giving subbands LL2 (8), HL2 (9), LH2 (10), HH2 (11). This process is continued for example as illustrated in Figure 3 wherein the LL4 subband is illustrated. Obviously, further levels of decomposition can be provided depending on the size of the input image. The lowest frequency subband is referred to as the DC subband. In the case of Figure 3, the DC subband is the LL4 subband.

Each single level DWT can, in turn, be inverted to obtain the original image. Thus, a J-level DWT can be inverted as a series of J-single level inverse DWT's.

To code an image hierarchically the DC subband is coded first. Then, the remaining subbands are coded in order of decreasing level. That is for a 4 level DWT, the subbands at level 4 are coded after the DC subband (LL4). That is the HL4, LH4 and HH4 subbands. The subbands at level 3 (HL3, LH3, and HH3) are then coded, followed by those at level 2 (HL2, LH2 and HH2) and then level 1 (HL1, LH1 and HH1).

With standard images, the encoded subbands normally contain the "detail" information in an image. After quantisation of the subbands, they often consist of a sparse array of values and substantial compression can be achieved by efficient encoding of their sparse matrix form.

Turning now to Fig. 4, there is shown the tiling of the subbands, such as HH1. The subbands are preferably tiles 410, 420, 430, 440 and 450 with 32x32 blocks of

coefficients beginning from the top left-hand corner. The nomenclature 32x32 refers to 32 rows by 32 columns respectively.

Before proceeding with a description of the embodiments, a brief review of terminology used hereinafter is provided. For a binary integer representation of a number, "bit n" or "bit number n" refers to the binary digit n places to the left of the least significant bit (beginning with bit 0). For example, assuming an 8-bit binary representation, the decimal number 9 is represented as 00001001. In this number, bit 3 is equal to 1, while bits 2, 1, and 0 are equal to 0, 0, and 1, respectively. In addition, a transform of an image may be represented as a matrix having coefficients arranged in rows and columns, with each coefficient represented by a bit sequence. Conceptually speaking the matrix may be regarded as having three dimensions; one dimension in the row direction; a second dimension in the column direction and a third dimension in the bit sequence direction. A plane in this three-dimensional space that passes through each bit sequence at the same bitnumber is referred to as a "bitplane" or "bit plane". The term "bit plane number n" refers to that bit plane that passes through bit number n.

To simplify the description and not to obscure unnecessarily the invention, the transform coefficients are assumed hereinafter to be represented in a fixed point unsigned binary integer format, with an additional single sign bit. Preferably, 16 bits is used. That is, the decimal numbers -9 and 9 are represented with the same bit sequence, namely 1001, with the former having a sign bit equal to 1 to indicate a negative value, and the latter having a sign bit equal to 0 to indicate a positive value. In using an integer representation, the coefficients are implicitly already quantized to the nearest integer value, although this is not necessary for embodiments of the invention. Further, for the purpose of compression, any information contained in fractional bits is normally ignored.

A region of an image frame includes a set of contiguous image coefficients. The term coefficient is used hereinafter interchangeably with pixel, however, as will be well understood by a person skilled in the art, the former is typically used to refer to pixels in a transform domain (eg., a DWT domain). These sets or regions T are defined as having transform image coefficients $\{c_{i,j}\}$, where (i,j) is a coefficient coordinate. fixed point unsigned binary integer format, with an additional single sign bit. We typically use 16 bits.

A set or the region T of pixels at a current bit plane is said to be insignificant if the msb number of each coefficient in the region is less than the value of the current bit plane. To make the concept of region significance precise, a mathematical definition is given in Equation (1). A set or region T of pixels is said to be insignificant with respect to (or at) bit plane n if,

$$|c_{i,j}| < 2^n, \text{ for all } c_{i,j} \in T \qquad (1)$$

APPENDIX A

By a partition of a set T of coordinates we mean a collection $\{T_m\}$ of subsets of T such that

$$T = \bigcup_{m=1}^{n} T_m, \qquad T_n \cap T_m = 0 \qquad \forall\, n \neq m$$

In other words if $c_{i,j} \in T$ then $c_{i,j} \in T_m$ for one, and only one, of the subsets T_m. In our case T is a square region and the set $\{T_m\}$ is the set consisting of the four quadrants of T.

The proposed method encodes a set of coefficients in an embedded manner using quadtrees. The use of the term embedded is taken to mean that every bit in a higher bit plane is coded before any bit in a lower bit plane. For example, every bit is coded in bit plane 7 before any bit in bit plane 6. In turn, all bits in bit plane 6 are coded before any bit plane 5 and so on.

A preferred embodiment of the proposed method is implemented utilizing the following pseudo-code. The proposed method preferably encodes a square block of coefficients, with a block size that is a power of 2 (typically 32x32 coefficients). Further, the proposed method utilizes a quadtree partition: that is each set or region is partitioned into its 4 quadrants: thus maintaining at all times square regions with a dimension equal to a power of two. The proposed method, during commencement, initializes three lists: a list of insignificant regions (LIR); a list of insignificant coefficients (LIC); and a list of significant coefficients (LSC). When single coefficients are removed from the list of insignificant sets (LIR), they are added to either the list of insignificant coefficients (LIC) or to the list of significant coefficients (LSC), depending on the significance of the coefficient.

The proposed method is initialized as follows. The LIC and LSC are initialized to be empty. The LIR is set to contain the four quadrants of the input block. The method commences by finding and coding $n_{\max}$, which is the largest bit plane that contains a 1 bit for all the coefficients in the set. The proposed method then proceeds as follows:

1. Set $n = n_{\max}$
2. For each coefficient in the list of insignificant coefficients (LIC)
   - Code bit n of the coefficient (i.e. its significance)
   - If the bit is 1 (i.e. it is significant) code a sign bit. Add the coefficient to the end of the LSC and remove the coefficient from the LIC.
3. For each region T in the list of insignificant regions (LIR)
   - Code the significance of T.
   - If T is significant and consists of more than one coefficient then partition T into its four quadrants and add these to the end of the LIR. Remove T from the list.
   - If T is a single coefficient
     - Remove T from the LIR

- If T is significant code a sign bit and add T to the end of the LSC
- Else add T to the end of the list of LIC

4. For each coefficient $c_{i,j}$ in the list of significant coefficients LSC (excluding those added to the list in step 3)
   - Code bit n of $c_{i,j}$.
5. decrement n and go to step 2.

From the above, it can seen that output bitstream generally takes the following form

...LIC'LIR'LSC'.....

where LIR' is the coded representation undertaken in step 3; LIC' is the coded representation undertaken in step 2; and LSC' is the coded representation undertaken in step 4. However, it should be noted that during the first iteration of the encoding process both LIC and LSC are empty and thus the output bitstream for the first iteration takes the form LIR'.

In addition to the proposed method, a simple Huffman code (or better a Golomb code) may be used to code groups of bits (for example groups of 4 bits) when coding the LIC and LSC. Further, when coding the significance of each quadrant of a region a 15-level Huffman code may be used to indicate the significance pattern of each of the 4 quadrants (one quadrant must be significant, hence the significance pattern can be one of 15 (and not 16) different patterns. Other forms of entropy encoding can be used, such as binary arithmetric coding to exploit any remaining redundancy.

As an alternative embodiment, the proposed method at step 3, if T consists of a 2x2 block of coefficients, may perform the following substep. Immediately code and output the significance of each coefficient of the 2x2 block, output the corresponding sign bit(s) if they are significant; and then add the coefficients to the end of the LSC or the LIC as appropriate. In the latter substep, the significant coefficients are added to the LSC list whereas the insignificant coefficients are added to the LIC list.

Preferably, the proposed method encodes a 32x32 block of data coefficients. For illustrative purposes only, the following example of a 4x4 block of coefficients is encoded in accordance with the proposed method.

$$\begin{bmatrix} 31 & 16 & 0 & 0 \\ 15 & 17 & 0 & 0 \\ 9 & 7 & 1 & 0 \\ 5 & 3 & 1 & 0 \end{bmatrix}$$

This Page Blank (uspto)

The above block consists of four quadrants A,B,C and D. The symbol A designates the top-left (2x2) quadrant of the block, B the top right, C the bottom left, and D the bottom right quadrant respectively. Furthermore, the symbols A1 denote the top left pixel of A, A2 the top right, A3 the bottom left, A4 the bottom right pixels respectively. Similarly B1 denotes the top left pixel of B and so on for the rest of the pixels.

According to the proposed method, n_{max} is first determined, which in this case is 4. That is, the most significant bit of each coefficient is in bit plane 4 or less. Note, the numbering of the bit planes commences from 0. The variable n_{max} is coded with 4 bits (since the coefficients have been constrained, so that n_{max} is between 0 and 15.). Initially

$$LIC = \phi, LIR = \{A, B, C, D\} \text{ and } LSC = \phi$$

where symbol ϕ is used to denote the empty list.

Then, according to the proposed method, the bit planes are iteratively coded. The process commences at bit plane $n = n_{max} = 4$, and decrements n by one at each iteration.

1. *At* $n = n_{max} = 4$

- First, each coefficient in the list LIC is coded. Since there are none, no coding is undertaken.
- Then, the significance of each region in the list LIR is coded.
  - For region A, a 1 bit is outputted, since it is significant at bit plane $n = 4$. Then, the four quadrants of A are added, namely A1, A2, A3 and A4, to the end of the list LIR, and A is removed. Hence now LIR = {B, C, D, A1, A2, A3, A4}.
  - For region B, a 0 bit is output, since it is insignificant at bit plane $n = 4$.
  - For region C, a 0 bit is output.
  - For region D, a 0 bit is output.
  - For region A1, a 1 bit is output. Since A1 consists of the single coefficient 31, A1 is removed from the LIR. Since 31 (or A1) is significant, it is added (or its location in the block) to the LSC. The sign bit (0) of A1 is also outputted.
  - For region A2, a 1 bit is output. Since A2 consists of the single significant coefficient 16, it is removed from the LIR, and added to the end of the LSC. The sign bit (0) of A2 is also outputted. Now we have LSC = {31, 16}.
  - For region A3, a 0 bit is output. Since it is a single insignificant coefficient we remove it from the LIR, add the coefficient 15 to the LIC. Now LIC = {15}.
  - For region A4 a 1 bit is output. Since A4 consists of the single significant coefficient 17, it is removed from the LIR, and added to the end of the LSC. The sign bit (0) of A4 is also outputted. Now LSC = {31, 16, 17}.
- Each coefficient in the LSC that was not added in the last step is now coded. Since there are none, no coding is undertaken.

Thus at the first iteration, the proposed method outputs the following bitstream

1000 10 10 0 10

At this stage, all the bits in bit plane 4 (and higher) have been coded. That is a decoder can reconstruct bit plane 4 (and higher) by reading in the bits from the coded bit stream. The decoding method is the same except that the significance decisions are determined by reading from the bit stream (this is why the significance decision is written to the bit stream). The other coefficient bits are simply read in as is. Note that the decoder execution path is identical to the encoder, so that the decoder knows the meaning of each new bit that it reads.

2. *At* n = *3*

Initially LIC = {15}, LIR = {B, C, D} and LSC = {31, 16, 17}.

- Firstly, bit *n*=3 of each coefficient in the LIC is coded. That is, a 1 bit is output for the coefficient 15 and a sign bit (0). Since it is significant (a 1 bit has been outputted), a sign bit is outputted, the coefficient 15 is removed from LIC and added to the end of the LSC. So now LSC = {31, 16, 17, 15}.
- The significance of each of the regions in LIR are now coded
  - For region B, a 0 bit is output.
  - For region C, a 1 bit is output, since it is significant at bitplane n=3. The region C is partitioned into four quadrants C00,C01,C10 andC11 which are added to the end of LIR. C is then removed from LIR. Hence now LIR = {B,D, C00,C01,C10, C11}.
  - For region D, a 0 bit is output.
  - For region C00, a 1 bit is output. Since C00 consists of the single significant coefficient 9, it is removed from the LIR, and added to the end of the LSC. The sign bit (0) of C00 is also outputted. Now we have LSC = {31, 16,17,15,9}.
  - For region C01, a 0 bit is output. Since it is a single insignificant coefficient we remove it from the LIR, add the coefficient 7 to the LIC. Now LIC = {7}.
  - For region C10, a 0 bit is output. Since it is a single insignificant coefficient we remove it from the LIR, add the coefficient 5 to the LIC. Now LIC = {7,5}.
  - For region C11, a 0 bit is output. Since it is a single insignificant coefficient we remove it from the LIR, add the coefficient 3 to the LIC. Now LIC = {7,5,3}
- Now we code bit *n*=3 of each coefficient on the LSC (that was not just added above)
  - We output 1, 0, and 0 as bit *n*=3 of 31, 16 and 17 respectively

Thus at the second iteration, the proposed method outputs the following bitstream

10 0 1 0 10 0 0 0 1 0 0

3. *At* n = *2*

Initially we have LIC = {7, 5, 3}, LIR = {C, D} and LSC = {31, 16, 17, 15, 9}.

- Firstly, bit n=2 (or equivalently the significance at bit plane n=2) of each coefficient in the LIC is coded. That is, we output a 1, 1, and 0 for 7, 5, and 3 respectively. In addition, a sign bit for 7 (0) and 5 (0) is outputted and these coefficients are moved to the LSC. We leave 3 in the LIC.
- Then the significance of each region in the LIR is coded
  - For region B, a 0 bit is output and for region D a 0 bit is output.
- Finally we update bit n=2 for each of the coefficients in the LSC (not added above).
  - We output a 1, 0, 0, 1, and 0 for 31, 16, 17, 15 and 9 respectively.

Thus at the third iteration, the proposed method outputs the following bitstream

10 10 0 0 0 1 0 0 1 0

We continue in this fashion until bit plane 0, or some other terminating point. Note that we can terminate after any one of the (three) sub-passes, if we use a special termination code. (Basically FF is reserved as a termination code, and we force the coded bit stream never to contain an FF, unless we deliberately insert a termination code.

As mentioned previously, the method is preferably utilized in encoding 32x32 blocks of coefficients. In these circumstances, the original quadrants A,B,C,D each consist of 16x16 coefficients and the regions A1,A2,...D4 each consist of 8x8 coefficients. It will be thus evident in encoding a 32x32 block, the block is partitioned in accordance with quadtree method five times, whereas in the example given the 4x4 block is partitioned only twice.

The decoding process simply mimics the encoding process to reconstruct the pixels from the coded representation.

Preferred Embodiment(s) of Apparatus.

Turning to Fig. 5, there is shown an encoder in accordance with a first preferred embodiment for implementing the proposed method. The coefficient encoder 500 is designed to provide a continual flow of output encoded data 502 taking in corresponding data 504. The encoder 500 includes the following logic portions; a bit plane converter 506; bit plane tree builders 508-0 to 508-15; and a bit plane encoder 510.

The encoder 500 also includes a memory 514 for storing the 32x32 coefficients in bit planes; a memory 516 for storing the sign bits of the coefficients; and memories 518-15 to 518-0 for storing bit plane trees 0 to 15. The encoder 500 further includes a memory 526 for storing a first list of regions; a FIFO memory 528 for storing a second list of regions; a memory 522 for storing a list of LSC values; and a memory 524 for storing a list of LIC values.

The encoder 500 operates in the following manner. Initially, the 32x32 input coefficient data are stored in the memory 512 in raster order. The bit plane converter 506 reads the input coefficient data and converts the data into 16 bit planes from bitplane 0

through to bitplane 15, which are subsequently stored in memory 514. The bit plane converter 506 also determines the sign bit of the coefficients of the 32x32 block, which sign bits are stored in memory 516.

The bit plane tree builders 508-0 to 508-15 each read the 32x32 coefficient data and construct a quadtree structure having nodes corresponding to regions and 1x1 pixels. In each tree, the nodes are set to 1 if the region or pixel corresponding to that node is significant for that bitplane. If the significant bit for the region or pixel corresponding to that node is greater than or less than the bitplane then the node is set to zero.

Turning now to Fig. 6, there is shown a constructed tree 700 built by a bitplane treebuilder 508-n at a bitplane n for a 32x32 block. For simplicities sake only a portion of the tree is shown. The tree 700 includes nodes representing each quadtree partition of the block down to the 1x1-pixel level. In this tree, the whole 32x32 block is represented by the symbol O. The 16x16 top left quadrant is represented by node A, the 16x16 top right quadrant is represented by B, the 16x16 bottom left quadrant is represented by node C, the 16x16 bottom right quadrant is represented by node D. The nodes A1,A2,A3,and A4 represent the top left, top right, bottom left, and bottom right 8x8 quadtree partitions of the quadrant A respectively. Similarly, the nodes A11, A12,A13,and A14 represent the top left, top right, bottom left, and bottom right 4x4 quadtree partitions of the quadrant A1 respectively. Similarly the nodes A111, A112,A113,and A114 represent the top left, top right, bottom left, and bottom right 2x2 quadtree partitions of the quadrant A11 respectively. Finally, the nodes A1111, A1112, A1113, and A1114 represent the top left, top right, bottom left, and bottom right 1x1 quadtree partitions (i.e. pixels) of the quadrant A111 respectively. The remaining parts of the tree (not shown) is represented in a similar manner down to each 1x1 quadrant of the 32x32 block.

The bit plane treebuilder 508-n builds such a tree from bottom up for bitplane n by reading the coefficients in quadrant order (e.g. A1111,A1112,A1113, and A1114). The tree builder sets the nodes of the tree to 1 if the region or pixel corresponding to that node is significant for that bitplane. If the significant bit for the region or pixel corresponding to that node is greater than or less than the bitplane then the node is set to zero by the tree builder. The bit plane tree builder then outputs the significance information for each node in the following format

A B C D A1 A2 A3 A4 A11 A12 A13 A14 A21,...,D44 A111,...,D444 A1111,...,D4444

The output from each of the bitplane tree builders 508-0 to 508-15 are then stored in respective bitplane tree memories 518-0 to 518-15.

The bit plane encoder 510 reads each of the bit plane tree memories in turn, commencing with bit plane tree memory 518-15. The bit plane encoder 510 starts

processing by reading in turn the four significance bits A,B,C, and D stored in the bit plane memory 518-15 corresponding to the nodes A, B, C, and D. The bit plane encoder stores a list of these nodes {A,B,C,D} in a first list of regions in memory 526. The bit plane encoder 510 then proceeds with the following operations:

1. The bit plane encoder 510 reads the bit in the bit plane tree corresponding to the first node (region) on the first region list.

a. If the bit is significant, the encoder outputs a binary one. The encoder then stores the children of the node in the second region list on the FIFO 528 and removes the node from the first region list 526.

b. If the bit is insignificant, the encoder outputs a binary zero and retains the node on the first region list 526.

2. The bit plane encoder 510 repeats step 1 until all nodes in the first region list have been read.

3.The bit plane encoder reads the bit in the bit plane tree corresponding to the first node in the second region list on the FIFO 528.

a. If the bit is insignificant and there are children to that node (viz. there are nodes directly below that node in the tree), the encoder outputs a binary zero and puts that node on first region list 526.

b. If the bit is insignificant and there are no children to that node, the encoder outputs a binary zero and that node is stored on the LIC list 524 as an index to the corresponding pixel.

c. If the bit is significant and has no children to that node, the encoder outputs a binary one and the corresponding sign bit and stores that node on the LSC list 522 as an index to the corresponding pixel.

d. If the bit is significant and has children to that node, the encoder outputs a binary one and removes the node from the second region list. In addition, it adds the children of that node to the second region list.

e. If the second region list is empty, the encoding process is completed for that bit plane tree.

The bit plane encoder repeats this operation for the remaining bit plane trees in turn. The first list of regions at the start of the operation on the current bitplane tree contains those regions remaining from the previous operation on the previous bitplane tree. These outputs bits correspond to the LIR' portion of the output stream. After completion of a current operation for a current bitplane tree, the bitplane encoder then encodes and outputs the LIC and LSC bits.

The bit plane encoder encodes the LIC bits by reading the LIC list for the index to the first pixel on the list, and using the current bit plane number selects the bit needed from the bit plane memory 514. If the selected bit is a binary zero, then the encoder

outputs a zero. If the selected bit is a binary one, then the encoder outputs a binary one together with the sign bit of the pixel from memory 516. The encoder then removes the index from the LIC list and adds it to the LSC list. Preferably, once an index is removed from the list, the remaining indices are reorganized. The encoding is completed once the LIC list is traversed.

The bit plane encoder encodes the LSC bits by reading the LSC list for the index to the first pixel on the list, and using the current bit plane number to select the bit needed from the bit plane memory 514. The selected bit is then outputted. The bit plane encoder also includes a counter for storing a length value, which is indicative of the number of pixels in the LSC list to be read. At the end of the LSC encoding the value is updated so that the new elements from LIR and LIC can be added.

Second Preferred Embodiment of Apparatus(s)

The encoding and decoding processes of the proposed method are preferably practiced using a conventional general-purpose computer, such as the one shown in Fig. 7, wherein the processes may be implemented as software executing on the computer. In particular, the steps of the coding and/or decoding methods are effected by instructions in the software that are carried out by the computer. The software may be divided into two separate parts; one part for carrying out the encoding and/or decoding methods; and another part to manage the user interface between the latter and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for encoding digital images and decoding coded representations of digital images in accordance with the embodiments of the invention.

The computer system 700 consists of the computer 702, a video display 716, and input devices 718, 720. In addition, the computer system 700 can have any of a number of other output devices including line printers, laser printers, plotters, and other reproduction devices connected to the computer 702. The computer system 700 can be connected to one or more other computers via a communication interface 708c using an appropriate communication channel 730 such as a modem communications path, a computer network, or the like. The computer network may include a local area network (LAN), a wide area network (WAN), an Intranet, and/or the Internet

The computer 702 itself consists of a central processing unit(s) (simply referred to as a processor hereinafter) 704, a memory 706 which may include random access memory (RAM) and read-only memory (ROM), input/output (IO) interfaces 708a, 708b

& 708c, a video interface 710, and one or more storage devices generally represented by a block 712 in Fig. 7. The storage device(s) 712 can consist of one or more of the following: a floppy disc, a hard disc drive, a magneto-optical disc drive, CD-ROM, magnetic tape or any other of a number of non-volatile storage devices well known to those skilled in the art. Each of the components 704 to 712 is typically connected to one or more of the other devices via a bus 714 that in turn can consist of data, address, and control buses.

The video interface 710 is connected to the video display 716 and provides video signals from the computer 702 for display on the video display 716. User input to operate the computer 702 can be provided by one or more input devices 708b. For example, an operator can use the keyboard 718 and/or a pointing device such as the mouse 720 to provide input to the computer 702.

The system 700 is simply provided for illustrative purposes and other configurations can be employed without departing from the scope and spirit of the invention. Exemplary computers on which the embodiment can be practiced include IBM-PC/ATs or compatibles, one of the Macintosh (TM) family of PCs, Sun Sparcstation (TM), or the like. The foregoing are merely exemplary of the types of computers with which the embodiments of the invention may be practiced. Typically, the processes of the embodiments, described hereinafter, are resident as software or a program recorded on a hard disk drive (generally depicted as block 712 in Fig. 7) as the computer readable medium, and read and controlled using the processor 704. Intermediate storage of the program and pixel data and any data fetched from the network may be accomplished using the semiconductor memory 706, possibly in concert with the hard disk drive 712.

In some instances, the program may be supplied to the user encoded on a CD-ROM or a floppy disk (both generally depicted by block 712), or alternatively could be read by the user from the network via a modem device connected to the computer, for example. Still further, the software can also be loaded into the computer system 700 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing are merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

The foregoing only describes a small number of embodiments of the present invention, however, modifications and/or changes can be made thereto by a person skilled in the art without departing from the scope and spirit of the invention. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

APPENDIX A

The following numbered paragraphs set forth aspects of the invention, including

1. An encoder for generating a coded representation of a digital image, said encoder including:

an input means for inputting a block of coefficients of said digital image;

a plurality of tree builders, wherein each tree builder generates a tree and nodes based on a corresponding bitplane of said block of coefficients, and each said node corresponds to one of a plurality of regions of said coefficients or to one of said coefficients and each said node having a data value indicative of the significance of said one region or said one coefficent for that bitplane;

a bitplane converter for converting the block of coefficients into their respective bitplanes; and

a bitplane encoder coupled to said plurality of tree builders and said bitplane converter, wherein said bitplane encoder using said trees and bitplanes outputs a coded representation of the bitplanes in sequence to produce a coded representation of the digital image.

2. An encoder as set forth in paragraph 1, wherein said encoder further includes: first storage means, coupled to said bitplane encoder, for temporarily storing a list of insignificant regions.

3. An encoder as set forth in paragraph 1 or 2, wherein said encoder further includes: second storage means, coupled to said bitplane encoder for temporarily storing a list of signficiant coefficents.

4. An encoder as set forth in any one of paragraphs 1 to 3, wherein said encoder further includes: third storage means, coupled to said bitplane encoder for temporarily storing a first list of significant regions.

5. An encoder as set forth in any one of paragraphs 1 to 4, wherein said encoder further includes: fourth storage means, coupled to said bitplane encoder for temporarily storing a second list of significant regions.

6. A method of generating a coded representation of a digital image.

7. A computer program product including a computer readable medium having recorded thereon a computer program for generating a coded representation of a digital image.

This Page Blank (uspto)

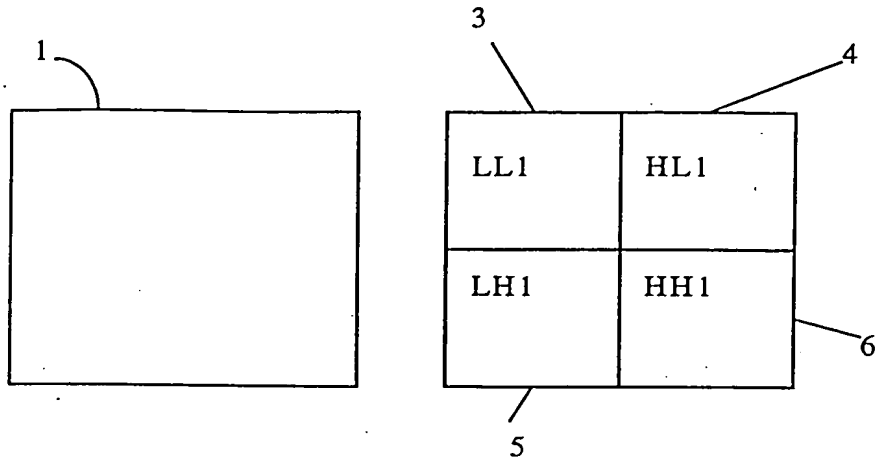


Figure1. One level DWT

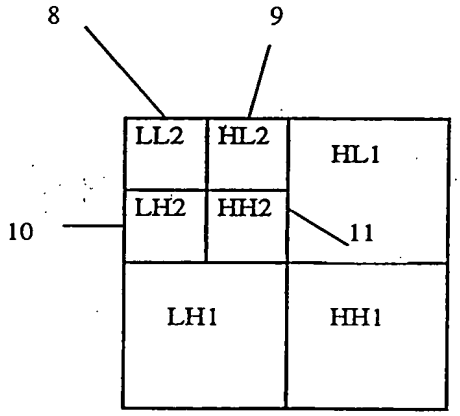


Figure 2. Two level DWT

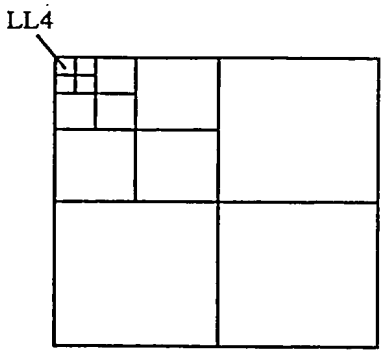


Figure 3. Four level DWT

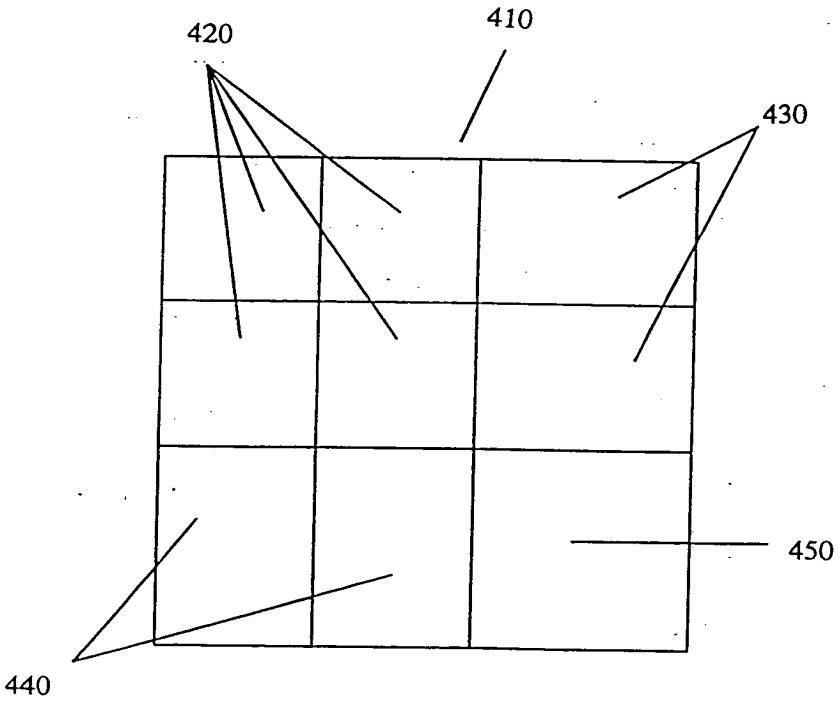


Figure 4. Tiling of the subbands

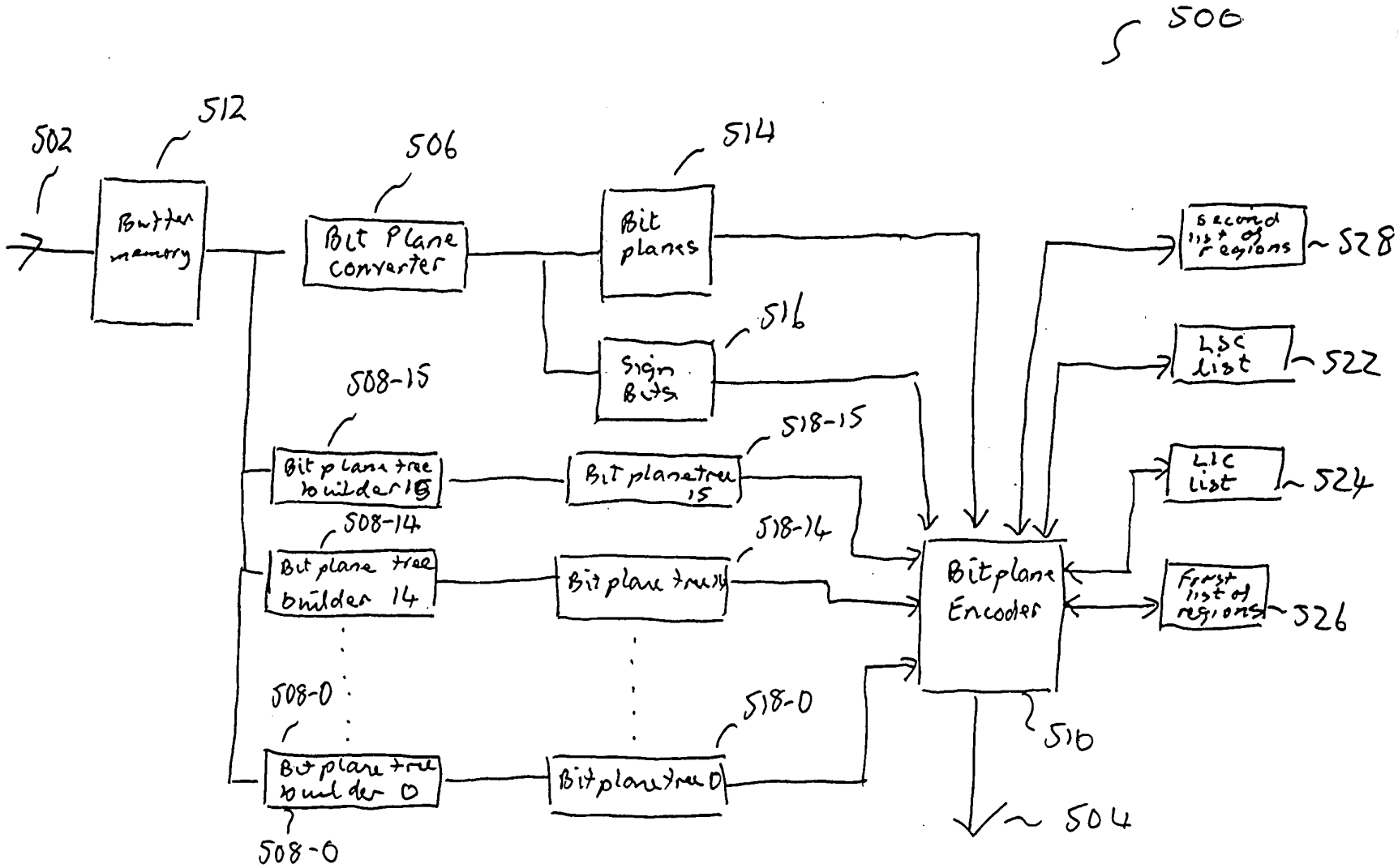

500
502
512
Buffer memory
506
Bit Plane converter
514
Bit planes
516
Sign Bits
508-15
Bit plane tree builder 15
518-15
Bit plane tree 15
508-14
Bit plane tree builder 14
518-14
Bit plane tree 14
508-0
Bit plane tree builder 0
518-0
Bit plane tree 0
508-0
Bitplane Encoder
510
504
Second list of regions
528
LSC list
522
LIC list
524
First list of regions
526

Fig 5

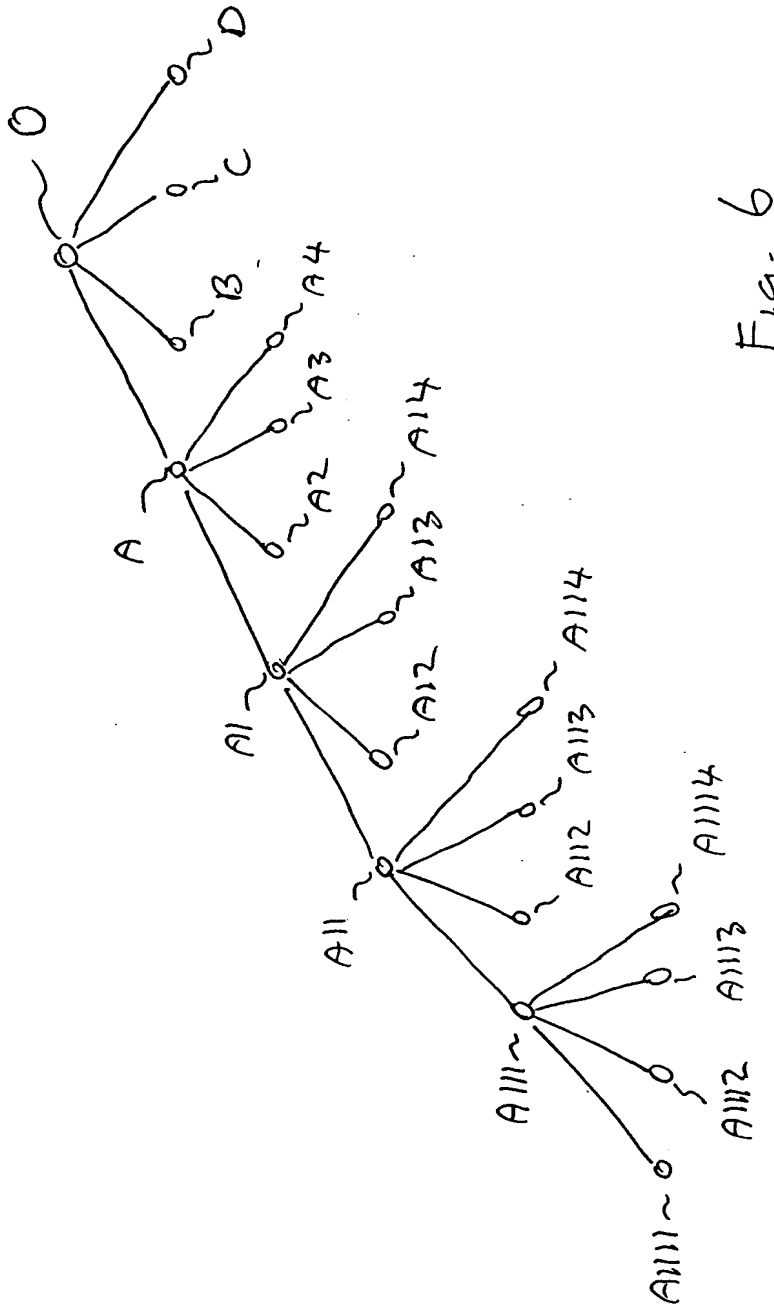

O
D
C
B
A4
A3
A2
A
A14
A13
A12
A1
A114
A113
A112
A11
A1114
A1113
A1112
A111
A1111

Fig. 6

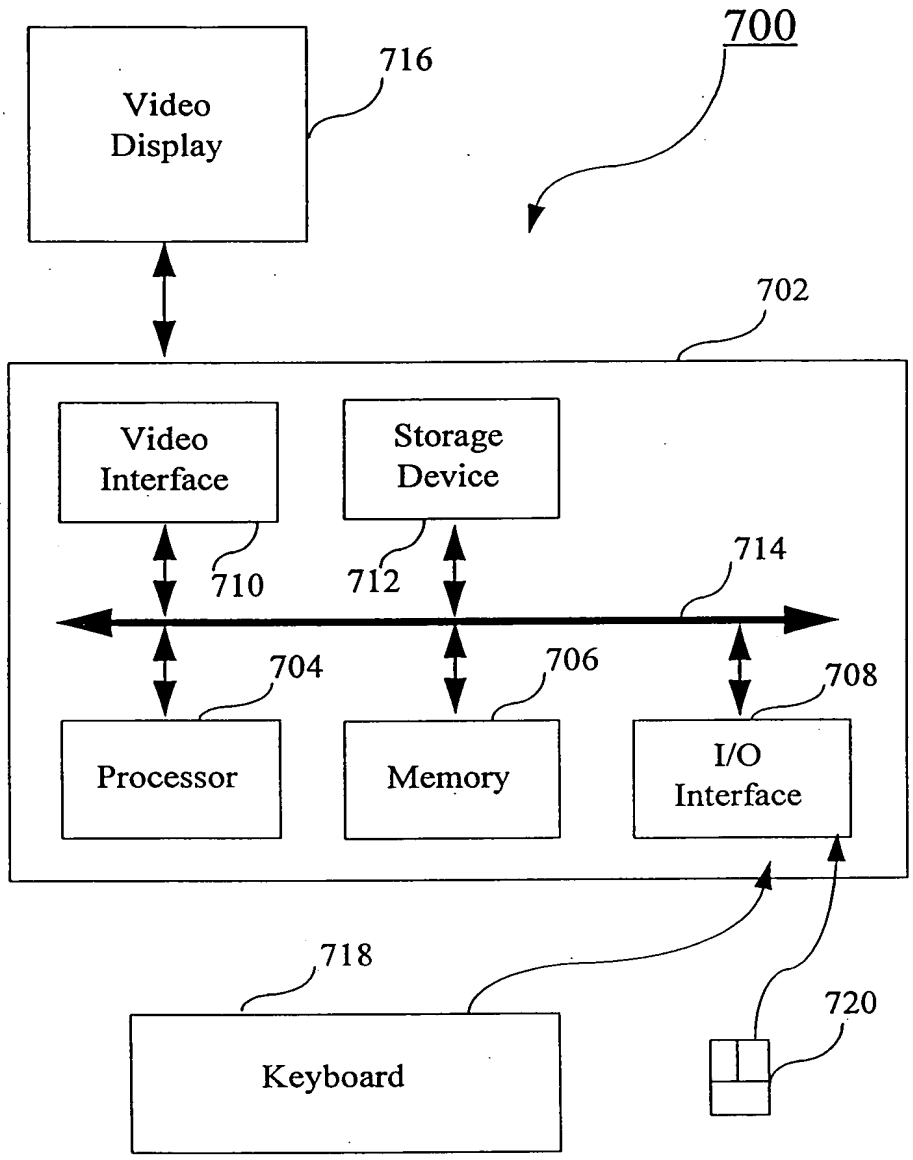

700
716
Video Display
702
Video Interface
Storage Device
710
712
714
704
706
708
Processor
Memory
I/O Interface
718
720
Keyboard

Fig. 7

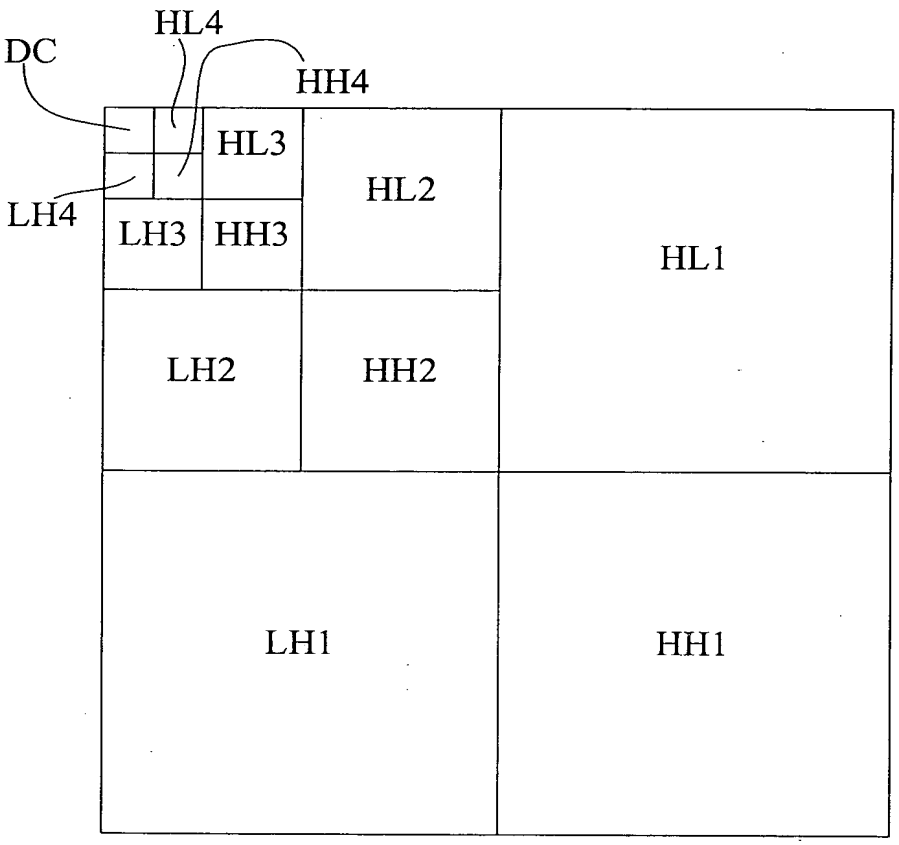


Fig. 1

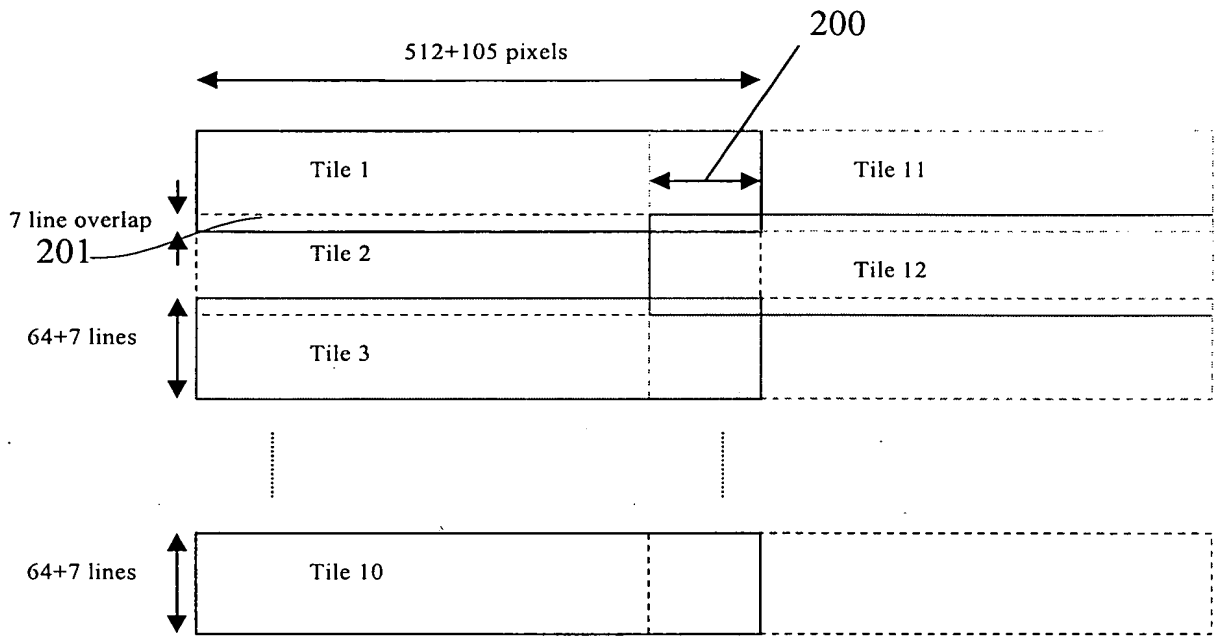

200
512+105 pixels
Tile 1
Tile 11
7 line overlap
201
Tile 2
Tile 12
64+7 lines
Tile 3
64+7 lines
Tile 10

Fig. 2

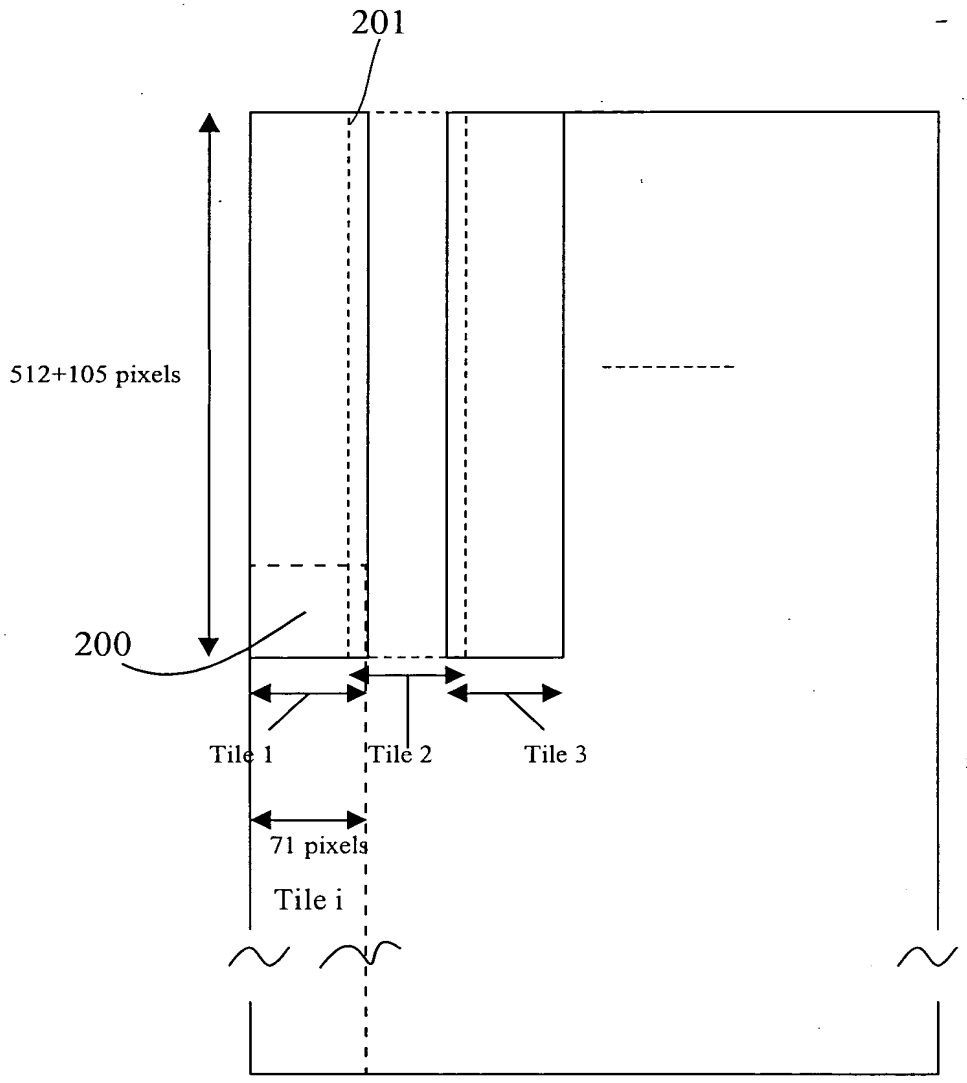


Fig. 3

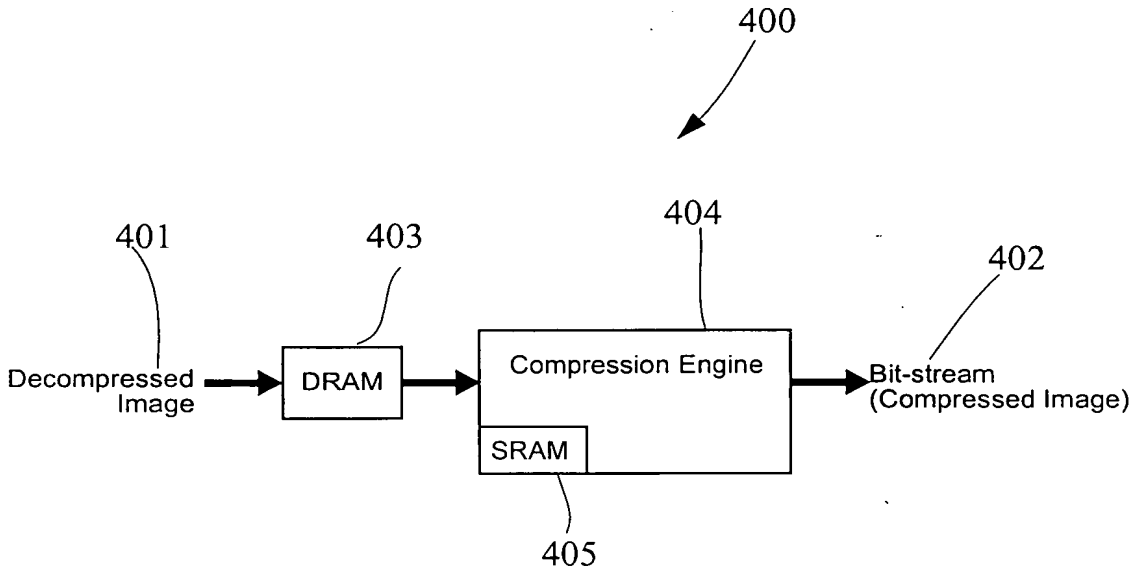

400
401
403
404
402
Decompressed Image
DRAM
Compression Engine
SRAM
Bit-stream (Compressed Image)
405

Fig. 4

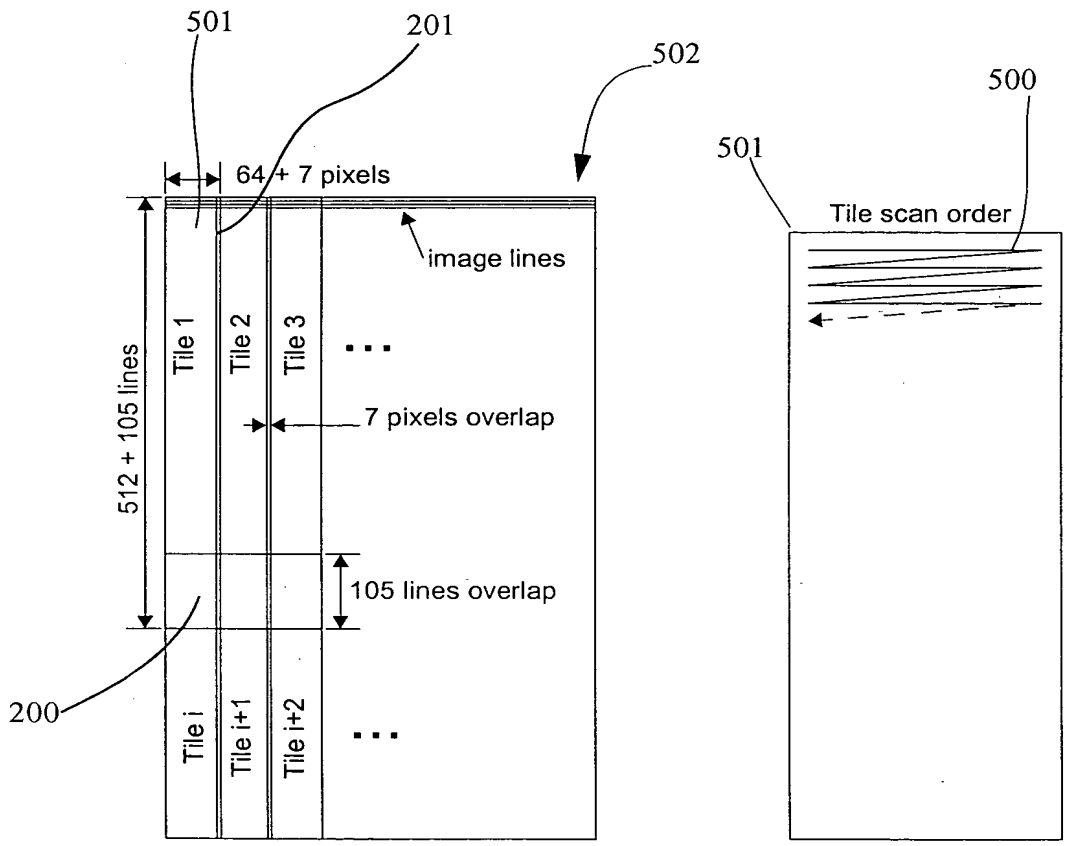

501
201
502
500
501
64 + 7 pixels
Tile scan order
image lines
Tile 1
Tile 2
Tile 3
...
512 + 105 lines
7 pixels overlap
105 lines overlap
200
Tile i
Tile i+1
Tile i+2
...

Fig. 5

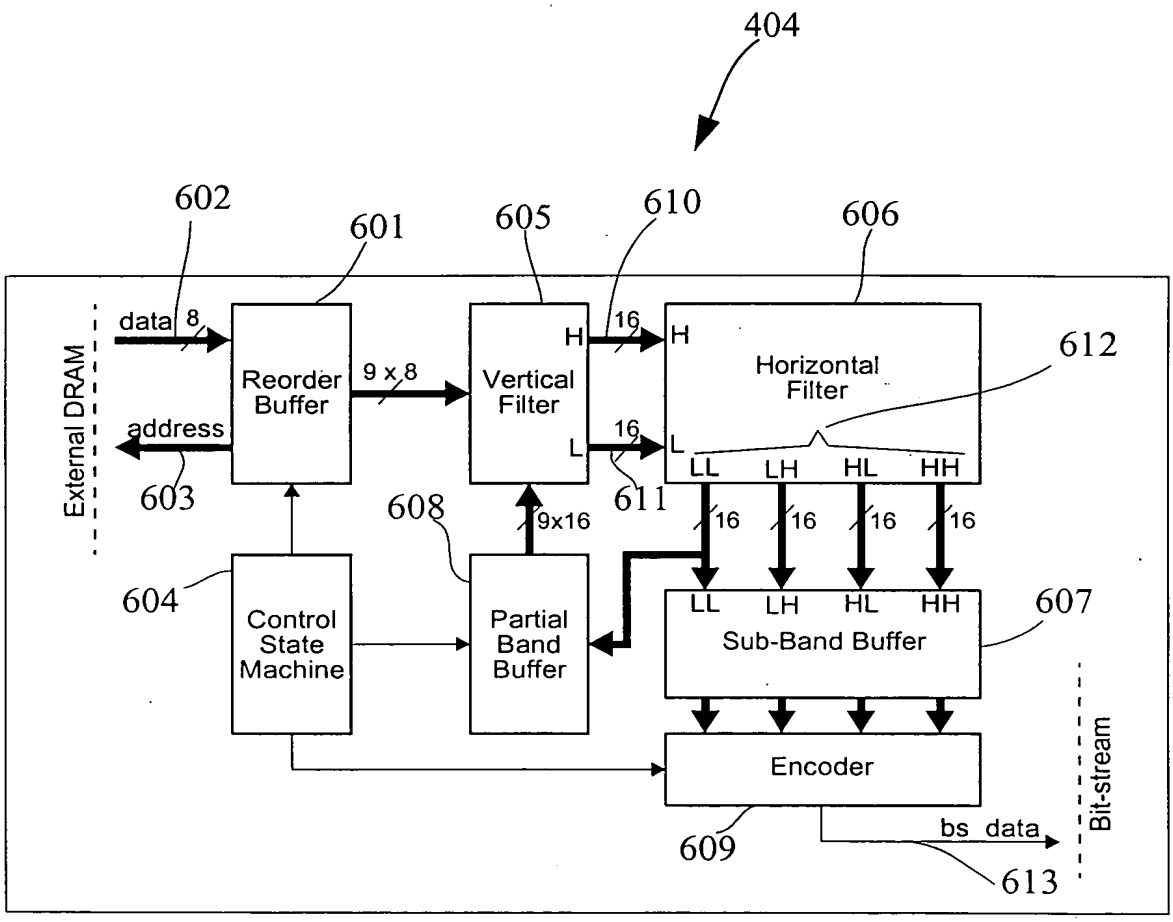

404
602
601
605
610
606
612
data
8
address
External DRAM
Reorder Buffer
9 x 8
Vertical Filter
H
L
16
16
Horizontal Filter
LL
LH
HL
HH
603
611
608
9x16
16
16
16
16
604
Control State Machine
Partial Band Buffer
LL
LH
HL
HH
Sub-Band Buffer
607
Encoder
609
bs data
613
Bit-stream

Fig. 6

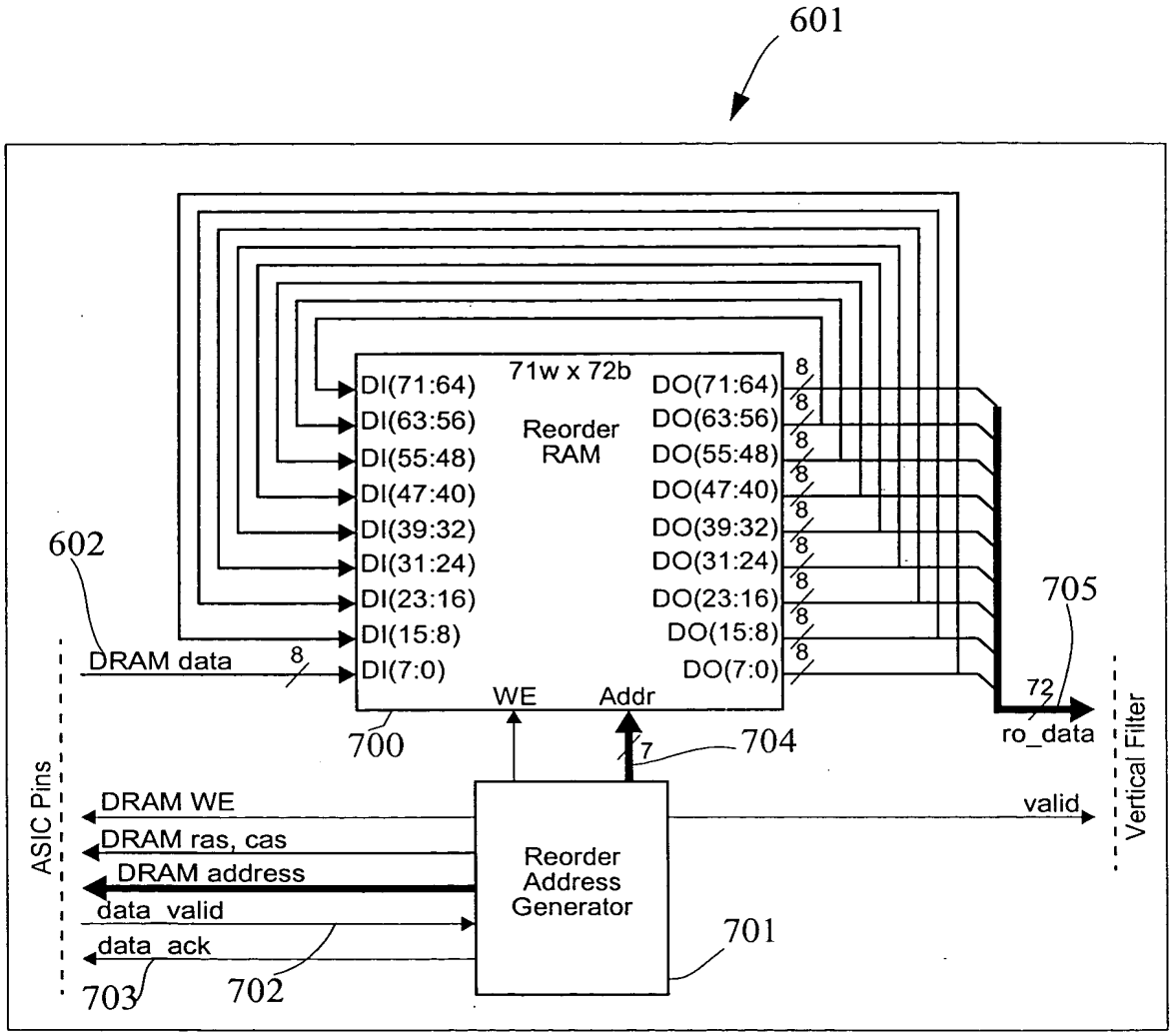

601
71w x 72b
Reorder RAM
DI(71:64)
DI(63:56)
DI(55:48)
DI(47:40)
DI(39:32)
DI(31:24)
DI(23:16)
DI(15:8)
DI(7:0)
DO(71:64)
DO(63:56)
DO(55:48)
DO(47:40)
DO(39:32)
DO(31:24)
DO(23:16)
DO(15:8)
DO(7:0)
8
602
DRAM data
WE
Addr
700
704
705
72
ro_data
Vertical Filter
ASIC Pins
DRAM WE
DRAM ras, cas
DRAM address
data_valid
data_ack
valid
Reorder Address Generator
701
702
703

Fig. 7

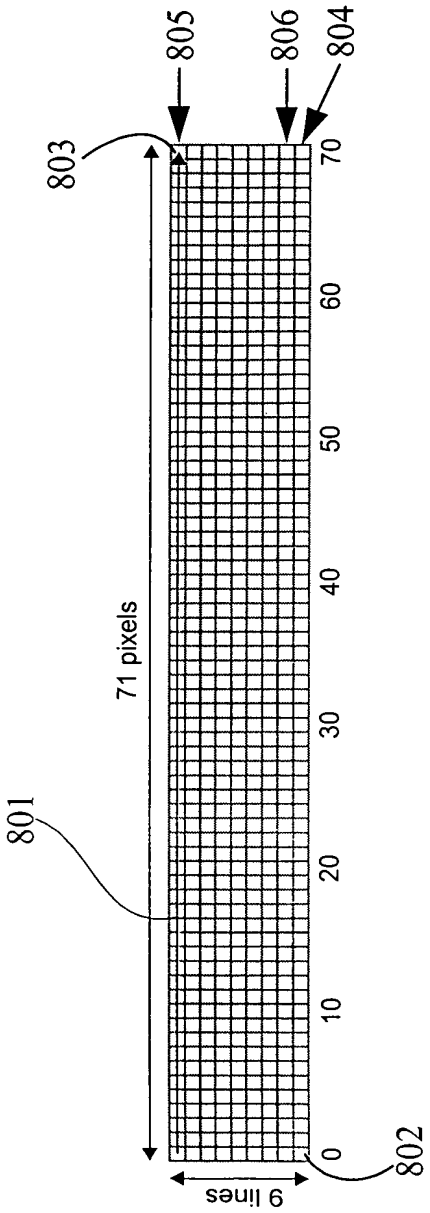


Fig. 8

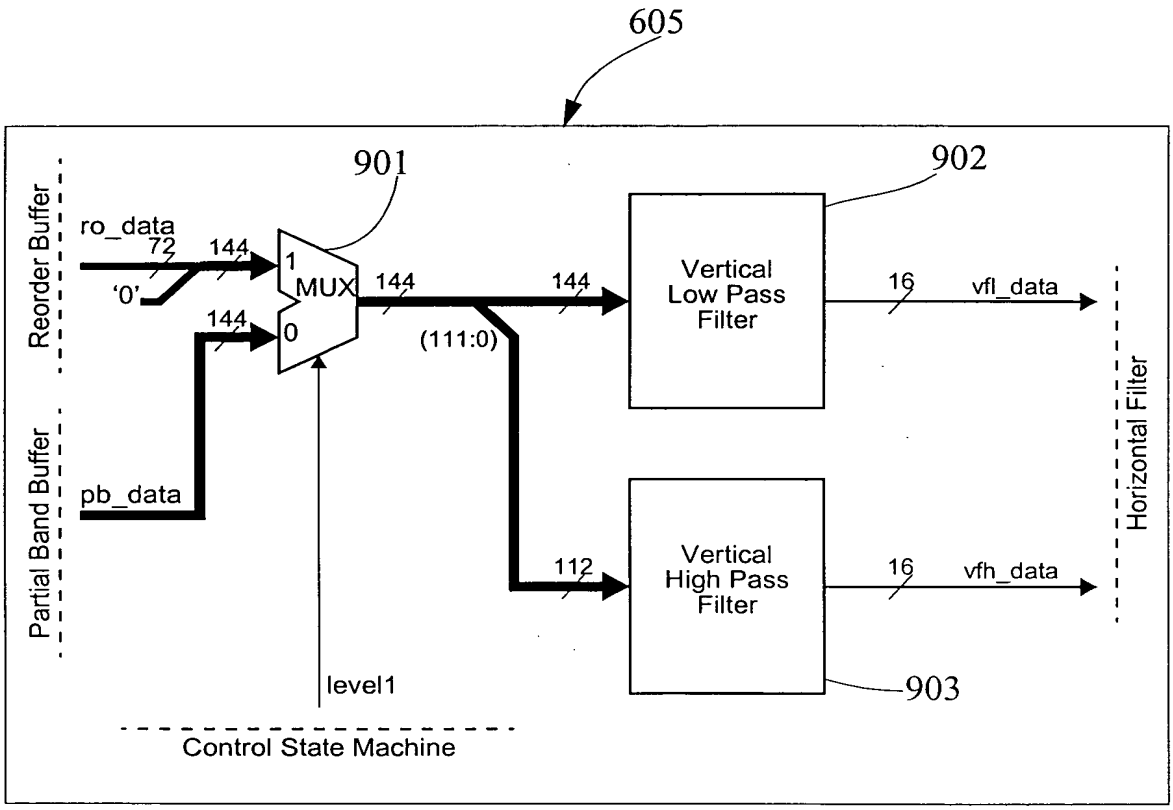

605
901
902
Reorder Buffer
ro_data
72
144
'0'
1
MUX
0
144
144
144
(111:0)
Vertical
Low Pass
Filter
16
vfl_data
Partial Band Buffer
pb_data
112
Vertical
High Pass
Filter
16
vfh_data
Horizontal Filter
level1
903
Control State Machine

Fig. 9

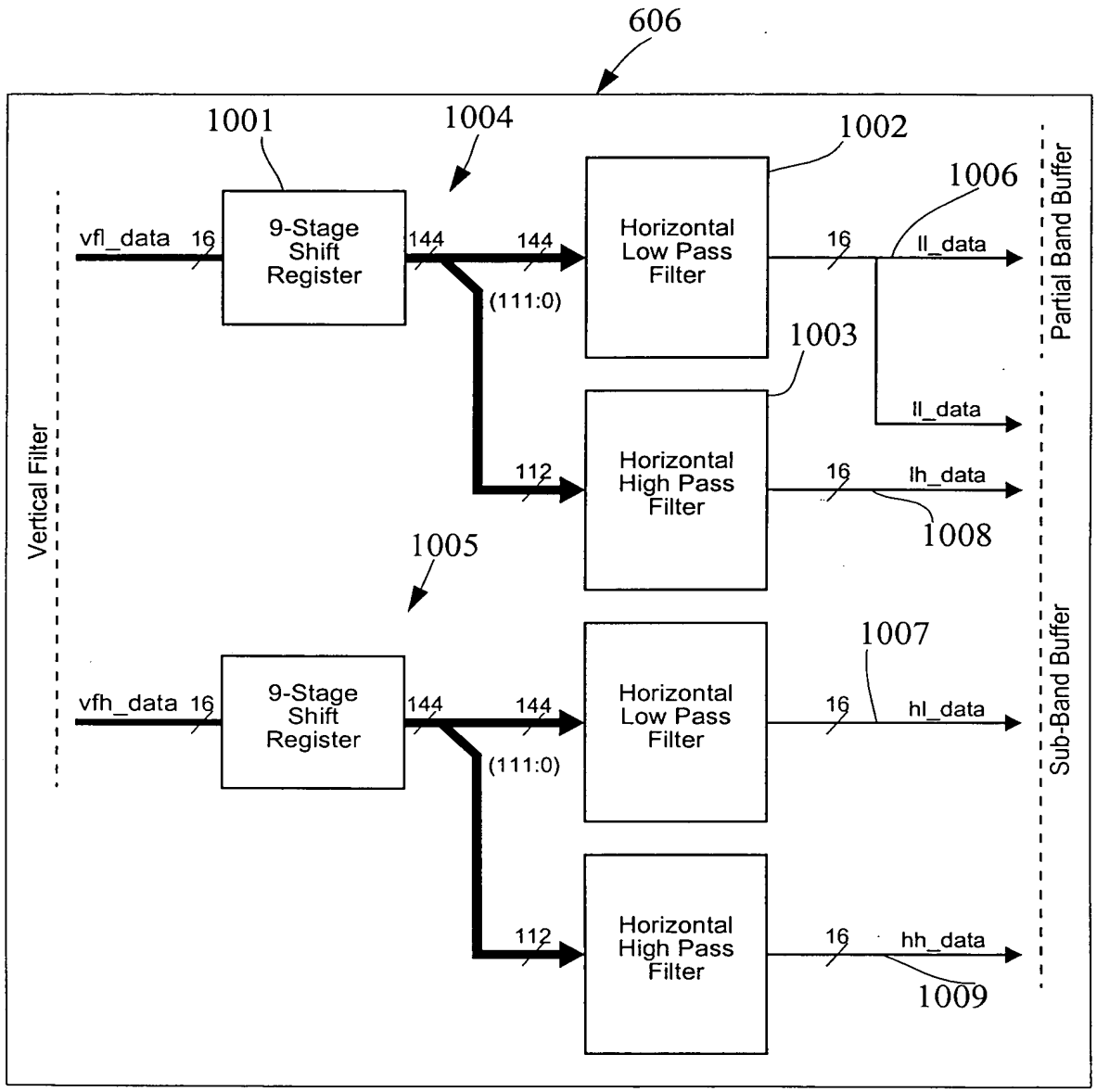

606
1001
1004
1002
1006
1003
1005
1007
1008
1009
vfl_data
16
9-Stage Shift Register
144
144
(111:0)
112
Horizontal Low Pass Filter
Horizontal High Pass Filter
16
ll_data
lh_data
hl_data
hh_data
Vertical Filter
Partial Band Buffer
Sub-Band Buffer
vfh_data
16
9-Stage Shift Register
144
144
(111:0)
112
Horizontal Low Pass Filter
Horizontal High Pass Filter
16
hl_data
16
hh_data

Fig. 10

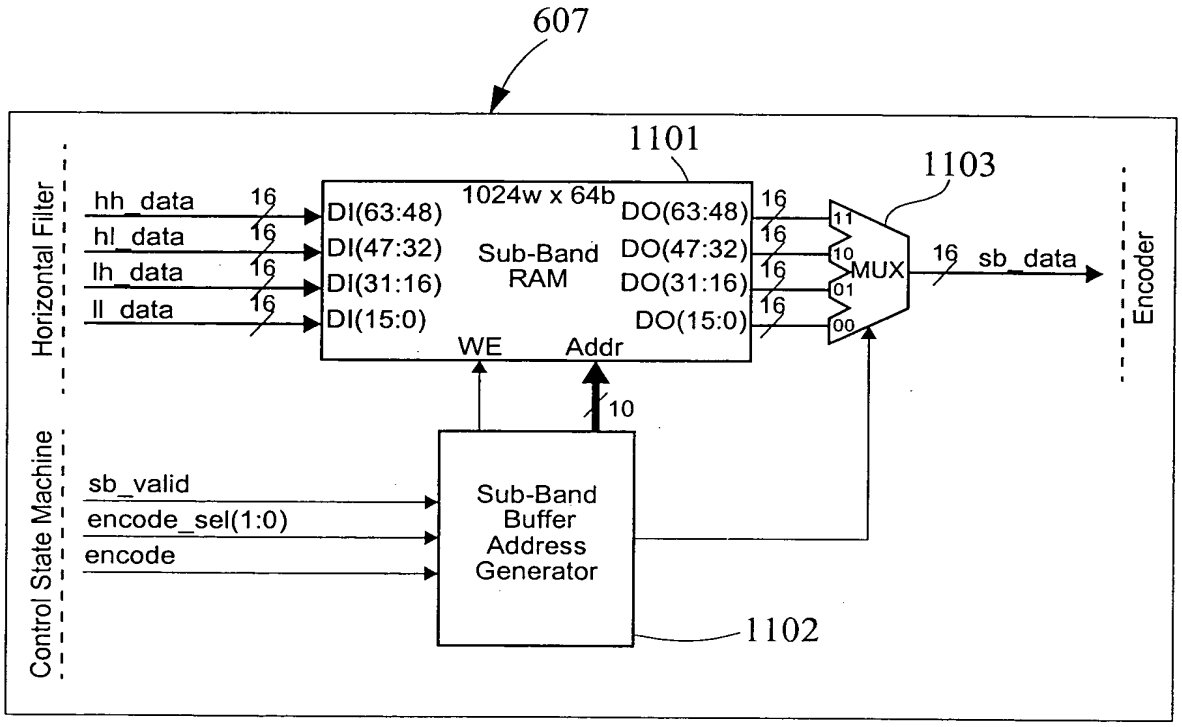

607
1101
1103
Horizontal Filter
Control State Machine
hh_data
hl_data
lh_data
ll_data
16
DI(63:48)
DI(47:32)
DI(31:16)
DI(15:0)
1024w x 64b
Sub-Band
RAM
DO(63:48)
DO(47:32)
DO(31:16)
DO(15:0)
WE
Addr
10
11
10
01
00
MUX
sb_data
Encoder
sb_valid
encode_sel(1:0)
encode
Sub-Band
Buffer
Address
Generator
1102

Fig. 11

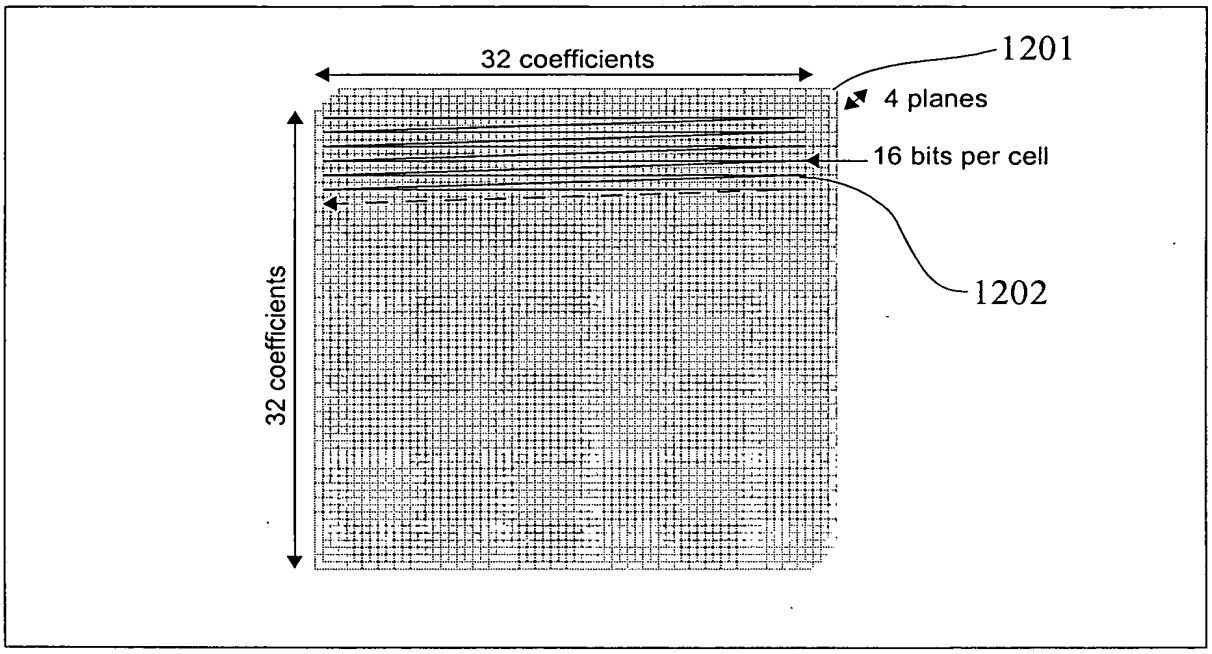

32 coefficients
1201
4 planes
16 bits per cell
1202
32 coefficients

Fig. 12

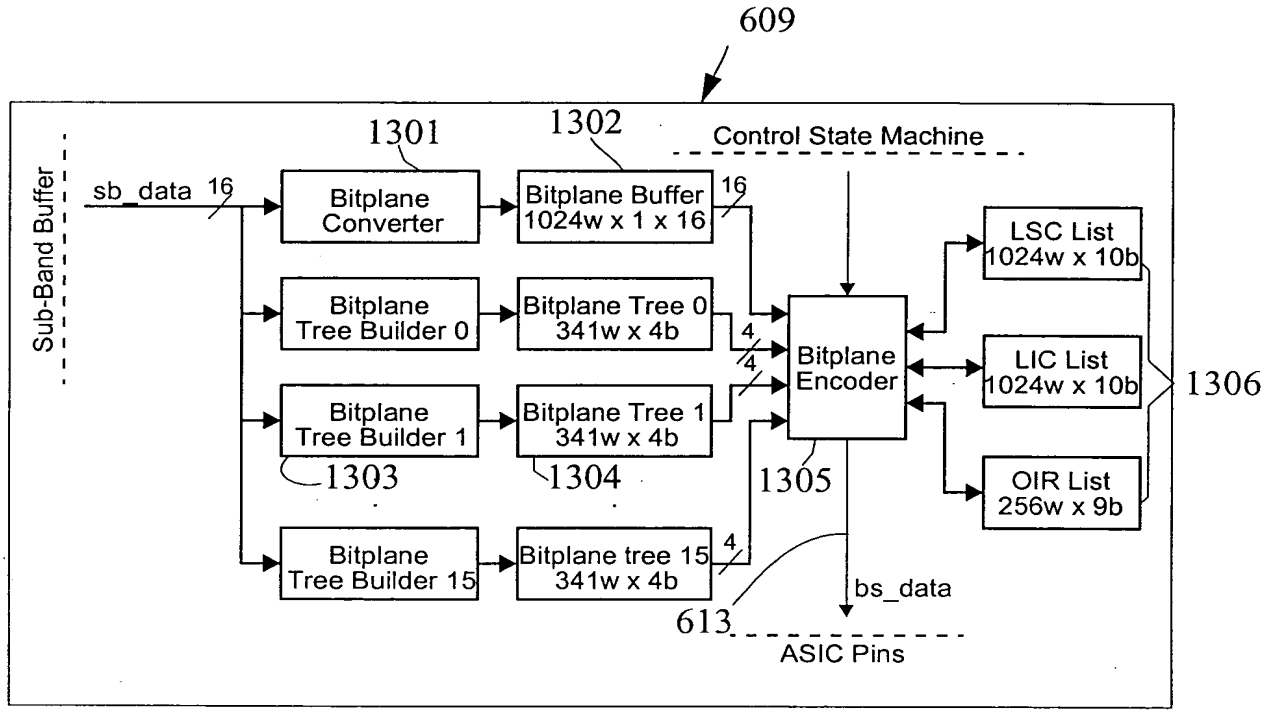

609
Sub-Band Buffer
sb_data
16
1301
Bitplane Converter
1302
Bitplane Buffer 1024w x 1 x 16
16
Control State Machine
Bitplane Tree Builder 0
Bitplane Tree 0 341w x 4b
4
Bitplane Tree Builder 1
Bitplane Tree 1 341w x 4b
4
1303
1304
Bitplane Encoder
1305
Bitplane Tree Builder 15
Bitplane tree 15 341w x 4b
4
LSC List 1024w x 10b
LIC List 1024w x 10b
OIR List 256w x 9b
1306
bs_data
613
ASIC Pins

Fig. 13

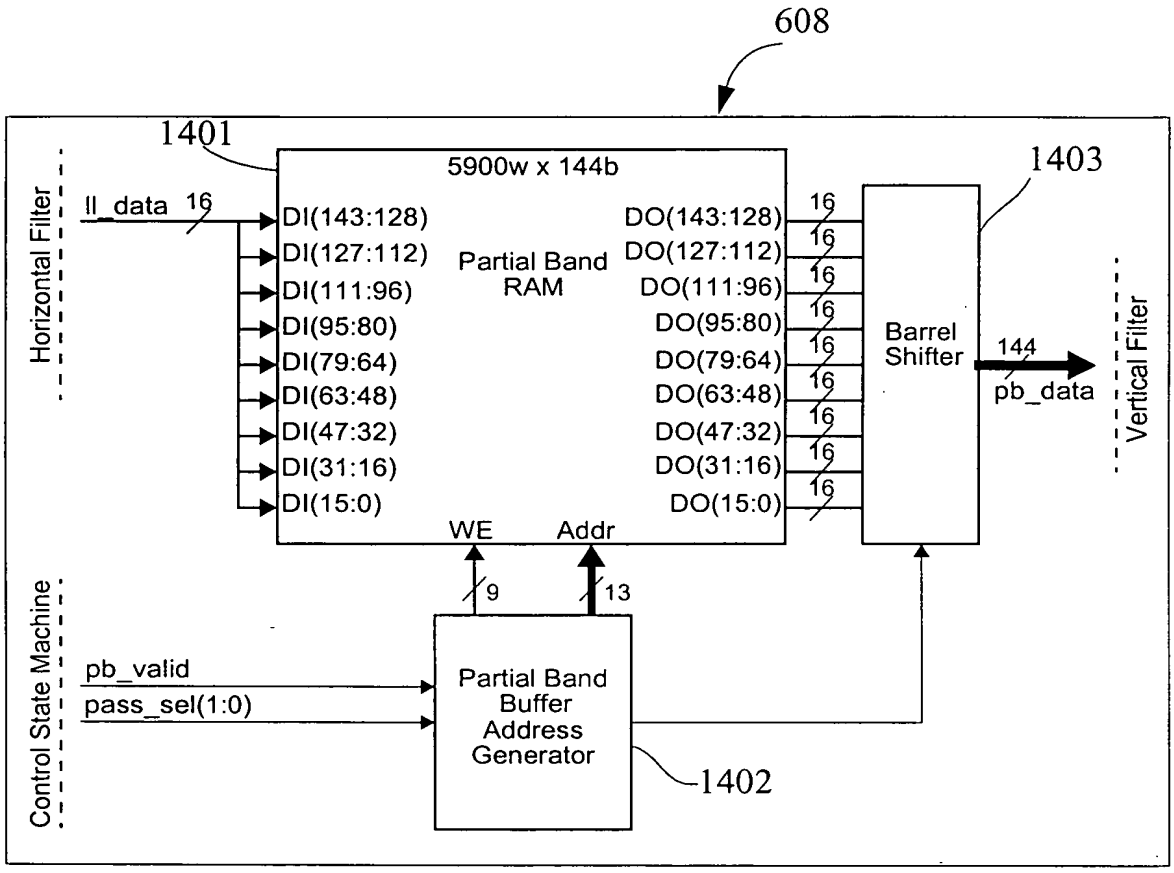

608
1401
5900w x 144b
1403
ll_data
16
DI(143:128)
DI(127:112)
DI(111:96)
DI(95:80)
DI(79:64)
DI(63:48)
DI(47:32)
DI(31:16)
DI(15:0)
Partial Band RAM
DO(143:128)
DO(127:112)
DO(111:96)
DO(95:80)
DO(79:64)
DO(63:48)
DO(47:32)
DO(31:16)
DO(15:0)
Barrel Shifter
144
pb_data
Horizontal Filter
Vertical Filter
WE
Addr
9
13
pb_valid
pass_sel(1:0)
Partial Band Buffer Address Generator
1402
Control State Machine

Fig. 14

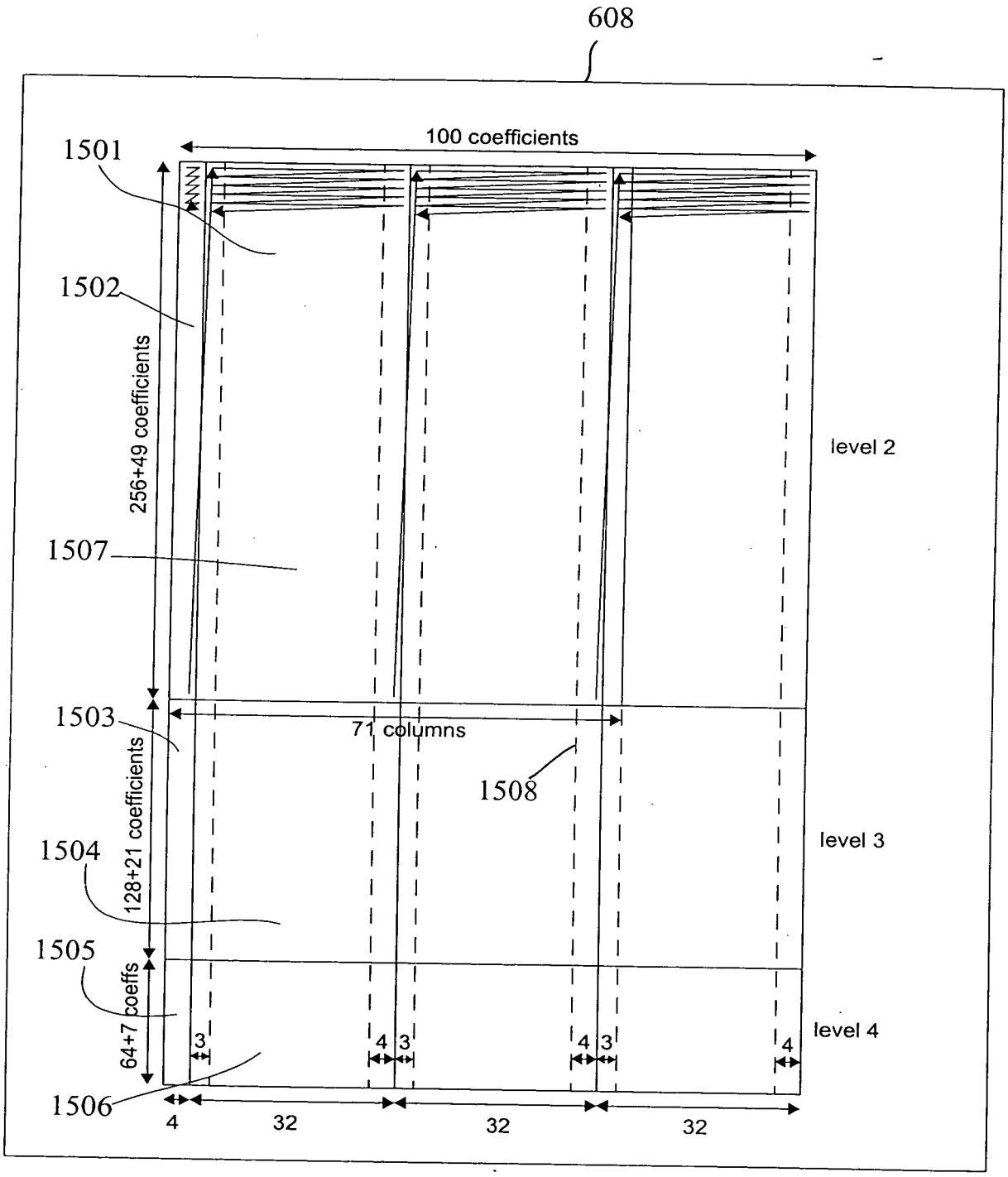

608
100 coefficients
1501
1502
256+49 coefficients
level 2
1507
1503
71 columns
1508
128+21 coefficients
level 3
1504
1505
64+7 coeffs
level 4
3
4
3
4
3
4
1506
4
32
32
32

Fig. 15

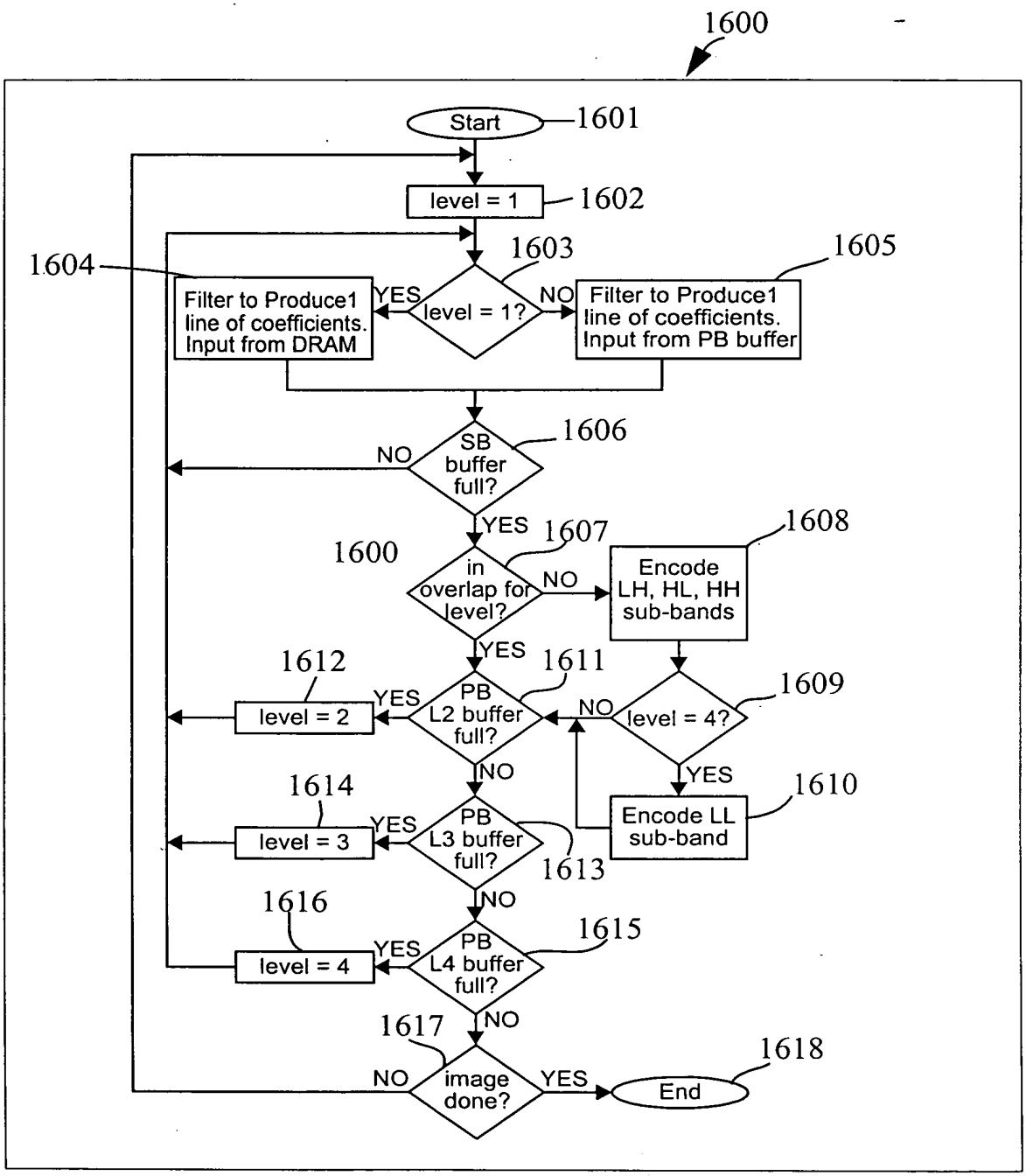

1600
Start
1601
level = 1
1602
1603
1604
1605
Filter to Produce1 line of coefficients. Input from DRAM
YES
level = 1?
NO
Filter to Produce1 line of coefficients. Input from PB buffer
1606
NO
SB buffer full?
YES
1600
1607
1608
in overlap for level?
NO
Encode LH, HL, HH sub-bands
YES
1612
1611
1609
level = 2
YES
PB L2 buffer full?
NO
level = 4?
NO
YES
1614
1610
level = 3
YES
PB L3 buffer full?
Encode LL sub-band
1613
NO
1616
1615
level = 4
YES
PB L4 buffer full?
NO
1617
1618
NO
image done?
YES
End

Fig. 16

This Page Blank (uspto)